Technical Report for NCC2-5202

IN-13

Autonomous Satellite Command and Control Through the World Wide Web – Phase 3

1 January 1997 – 30 September 1998

Stanford University
Space Systems Development Laboratory
Professor Brian Cantwell, Principal Investigator
Professor Robert Twiggs, Co-Investigator

1.0 Introduction

NASA's New Millenium Program (NMP) has identified a variety of revolutionary technologies that will support orders of magnitude improvements in the capabilities of spacecraft missions. This program's Autonomy team has focus on science and engineering automation technologies. In doing so, it has established a clear development roadmap specifying the experiments and demonstrations required to mature these technologies. The primary developmental thrusts of this roadmap are in the areas of remote agents, PI/operator interface, planning/scheduling fault management, and smart execution architectures.

Phases 1 and 2 of the ASSET Project (previously known as the WebSat project) have focused on establishing World Wide Web-based commanding and telemetry services as an advanced means of interfacing a spacecraft system with the PI and operators. Current automate capability includes Web-based command submission, limited contact scheduling, command list generation and transfer to the ground station, spacecraft support for demonstrations experiments, data transfer from the ground station back to the ASSET system, data archival, and Web-based telemetry distribution. Phase 2 was finished in December 1996.

Phase 3 of the ASSET Project was January-December 1997 and is the subject of this report. This phase permitted SSDL and its project partners to expand the ASSET system in a variety of ways. These added capabilities included the advancement of ground station capabilities, the adaptation of spacecraft on-board software, and the expansion of capabilities of the ASSET management algorithms. Specific goals of Phase 3 were:

(1) Extend Web-based goal-level commanding for both the payload PI and the spacecraft engineer.
(2) Support prioritized handling of multiple PIs as well as associated payload experimenters.
(3) Expand the number and types of experiments supported by the ASSET system and its associated spacecraft.
(4) Implement more advanced resource management, modeling and fault management capabilities that integrate the space and ground segments of the space system hardware.
(5) Implement a beacon monitoring test.
(6) Implement an experimental blackboard controller for space system management.
(7) Further define typical ground station developments required for Internet-based remote control and for full system automation of the PI-to-spacecraft link.

Each of those goals is examined in the next section. Significant sections of this report were also published as a conference paper [1].

2.0 Results

2.1 Web-Based Goal-Level Commanding

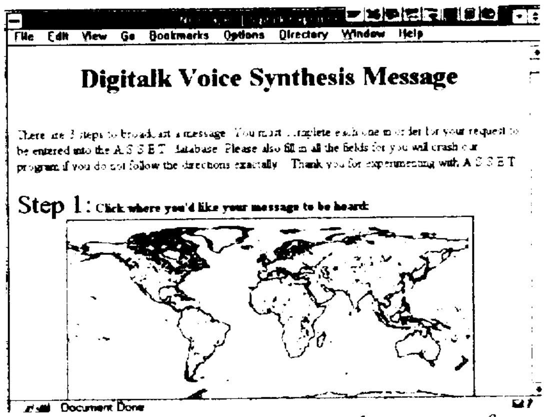
One of the goals of the ASSET system is to enable PIs and other customers to simply and effectively request services. The Phase 3 contributions to this goal were primarily conceptual, although the concepts were prototyped. As part of his PhD research, Christopher Kitts examined the high-level specification of products.

Requesting a product or service from a space system is typically an inefficient processs. First, requests are often directed to a human mission planner in a slow turn-around proposal process or in a manner that requires synchronization in time and/or space. Second, clients are often asked to specify their product at a level inappropriate to their knowledge or desire. Third, the process used to specify a desired product commonly overconstrains the set of possible implementations; this can severely limit the flexibility of the mission planning process when resources are constrained.

The ASSET system addresses these problems by incorporating a simple World Wide Web-based client interface capable of obtaining product requests at varying levels of abstraction. This interface decouples interaction with the mission control center so products may be specified at the convenience of the client. The interface leads the client through an interview process in order to gather information required to completely specify the product request. At the highest level, this process allows the client to provide only the pertinent attributes and delivery information concerning the product. At the lowest level, this process allows the client to specify specific implementation details or even the contact procedures themselves. Finally, the interface captures the client's product specification and submits it to the ASSET mission planning system. The specification is interpreted as the set of all implementations that can successfully produce the client's product.

In order to develop a client interface capable of both high-level abstraction and low-level detail, it was necessary to first develop a conceptual model of the products and services available to the user. The Object Role Modeling (ORM) technique was adopted to define the relationships between high-level and low-level conceptualizations of the product [2]. Using this model, a comprehensive client interface was developed.

The client interface was prototyped using an application written in Javascript and HTML [3]. A screen shot of one portion of the interview is shown in Figure 1. The interview application is mostly loaded on the client-side, except for the initial password authentication and the final submission. The output of this interview process is a model of the desired product which is sent by electronic mail to the main blackboard of ASSET. Further steps in product processing are described in Section 2.6.



Digitalk Voice Synthesis Message

There are 3 steps to broadcast a message. You must complete each one in order for your request to be entered into the A S S E T database. Please also fill in all the fields for you will crash our program if you do not follow the directions exactly. Thank you for experimenting with A S S E T

Step 1: Click where you'd like your message to be heard:

Figure 1: *The prototype client interface*

2.2 Handling of Multiple-PIs

Another goal of ASSET is to accommodate many clients at with different priority levels. For example, both "official" principal investigators and the amateur community should have access to products of the ASSET system, but amateur requests should not override the ability of scientists to accomplish their tasks.

ASSET satisfies the Phase 3 goal of incorporating multiple PIs through use of the client interface, described in Section 2.1. This client interface is accessible through the World Wide Web, making it widely available. The client requests are forwarded to the main ASSET scheduling system.

The goal of handling multiple levels of request authority is also satisfied through the client interface. High-priority users such as payload PIs and ASSET operators will have password access to the client interface; the password adds additional levels of priority to the product requests. ASSET incorporates three levels of priority: Priority 1 is for PIs and system operators; Priority 2 is for outside clients with special arrangements through ASSET; Priority 3 is for the general public. The scheduling system is defined such that no combination of lower-priority requests can ever prevent a higher-priority request from being scheduled. The scheduler is further discussed in Section 2.4

2.3 Expanded Experiment Set

Phase 3 also involved the expansion of the scope of the ASSET system to more experiments and spacecraft. The high-level conceptualization of the product model by Kitts, as described in Section 2.1, enables ASSET to incorporate almost any sort of experiment into the system. This was demonstrated by developing models for the Sapphire telemetry, voice and camera experiments.

The Sapphire flight software itself was also modified for use within ASSET. Specifically, the Chatterbox operating system was expanded to include health monitoring tables and a context-based beacon broadcast capability [4]. Health monitoring tables are an automated form of the telemetry limit checking process common to spacecraft operations; the Sapphire CPU reads the on-board sensors, comparing the values against safety thresholds. If a threshold is crossed, the

CPU acts to safe the vehicle and changes the state of the beacon [5]. The beacon experiment is further discussed in Section 2.5.

In addition to the changes in the Sapphire operating system during Phase 3, the Sapphire mission operations guide was modified to facilitate use in ASSET. The operations guide was written as a series of HTML documents, enabling it to be readily called up by human and automated operators.

Preliminary design of the operating system for Opal, SSDL's second spacecraft, was begun during Phase 3. ASSET developers contributed to the requirements definition process for the Opal Operating System to ensure that Opal can be operated through ASSET. Specifically, a beacon system functionally similar to Sapphire was incorporated into the requirements.

2.4 Advanced Capabilities

The Phase 3 goals for advanced capabilities were to develop and implement more advanced health management, resource scheduling and modeling methods.

As part of his PhD studies, Christopher Kitts examined health monitoring from a "first principles" approach. He has identified extensions to the contemporary fault management theory to cover anomalies that are not faults, and to define and automate the recovery task. For Phase 3, Kitts also developed the first draft of the health monitoring design document for Sapphire.

As part of his PhD studies, Michael Swartwout examined health monitoring from the perspective of observability. He has identified research opportunities to clarify and optimize the telemetry downlink, enhancing the operator's ability to perform health monitoring while reducing the required size of data [6].

As part of his PhD studies, Brian Engberg developed a prototype scheduler for ASSET. This automated scheduler works on a First Come, First Opportunity (FCFO) basis; the system attempts to schedule each request based on order of input. This basic scheduler incorporates the multiple priority levels discussed in Section 2.2 by maintaining three separate input lists. All of the Priority 1 users are scheduled, then the Priority 2, followed by the Priority 3.

This scheduler was demonstrated using standalone input and output files. During the next Phase of development, the scheduler will be directly incorporated into the ASSET UNIX environment.

2.5 Beacon Monitoring Experiment

One of the major goals for ASSET during Phase 3 was to begin the process of beacon monitoring experimentation. The ASSET team had the unique capability to run a controlled series of experiments to validate the use of automated health monitoring for Earth-orbiting spacecraft [7]. The tasks for Phase 3 included: developing the beacon monitoring design document, developing low-cost beacon receiving stations, improving the capabilities of Sapphire, and creating the ASSET beacon monitoring software.

During the early months of Phase 3, the first draft of the beacon monitoring design document was circulated to the participating institutions. By June 1997, there were three universities developing beacon monitoring stations for ASSET: Tuskegee University, Montana State University, and Sweden's Space Physics Institute. However, by September 1997, differences in program goals and the difficulty of communicating detailed designs forced SSDL to begin its own beacon receiving station development. Carlos Niederstrasser joined the ASSET team; his Engineer's Degree project is the development of the beacon receiving station.

Revisions to the Sapphire code began in the summer of 1997, as discussed in Section 2.3. The Sapphire flight software with beacon revisions was completed in June 1998. In addition, the Sapphire vehicle had to be modified to produce low-bandwidth beacon signals. Link studies demonstrated that the primary design choice of a packetized beacon would not have sufficient margin to be detected by an omnidirectional antenna. Instead, spacecraft designers took advantage of an unexpected operating mode of one of the payloads; when turned on, this device keys the transmitter carrier wave. Sapphire's beacon was defined as a set of bits with each bit corresponding to a different pulse length of the carrier.

Within ASSET, an automated process called Beaconwatch was developed to automatically handle incoming messages. The receiving stations are designed to send electronic mail with a translation of the received beacon message; Beaconwatch logs those messages in the ASSET database and automatically notifies operators when important messages are received.

2.6 Blackboard Controller

ASSET is envisioned as a fully automated system, integrating product requests, system scheduling, and analysis within one framework. The Phase 3 goal for integration was to establish the blackboard controller at the heart of ASSET. As shown in Figure 2, a blackboard architecture consists of a centralized database (the blackboard) which is manipulated by a set of knowledge experts. The ASSET architecture is arranged such that all the system data (such as system models, orbit windows, flight data, product requests) is stored in the blackboard. All of the specialized tasks of operations (such as planning, scheduling, contact control, orbit analysis, health management, product delivery) will be handled by independent experts, both human and automated.

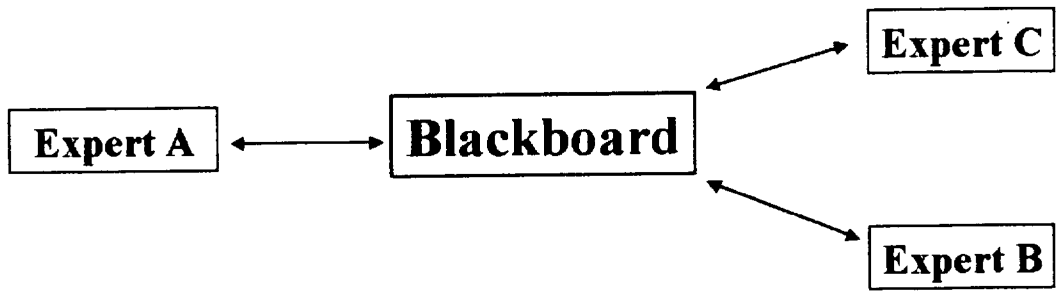


Figure 2: *Simple blackboard schematic*

The ASSET blackboard system is based on a freely-available application called BBK. BBK was developed by Barbara Hayes-Roth and Lee Brownston of Stanford's Knowledge Systems Laboratory. Given that this application is an experimental free product, one of the primary challenges during this Phase was to install and troubleshoot the program on SSDL computers. Mike Swartwout acted as the lead for learning and documenting BBK for use in ASSET.

During Phase 3, BBK was installed and successfully operated. The Beaconwatch process described in Section 2.5 was interfaced to the blackboard through a socket, allowing the two programs to share information. This final step enabled a partial demonstration of the beacon monitoring architecture in July 1997: An electronic mail beacon message was received by ASSET, converted into blackboard data, the message was interpreted as an alert, and an operator was notified.

As discussed in Section 2.2, the client interface forwards finished requests to the blackboard for processing. Another electronic mail processing application called Requestwatch was written to support this function. BBK was modified to allow multiple socket connections, and another partial demonstration was successful: a user requested a product through the client interface, which sent mail to the blackboard application, which converted the data into the blackboard and sent a confirmation message to the user.

2.7 Internet-Based Remote Control

The final goal for Phase 3 was to introduce internet-based remote control of ground station equipment. As part of his coursework, Jamie Cutler developed Mercury, a Windows-based application. Mercury enables remote users to log into the ground station using standard telnet protocols. The users can adjust all the radio and modem parameters as if they were physically present and run remote satellite contacts. Mercury runs as a background process on a PC, enabling users to run other applications at the same time.

Mercury was demonstrated with both the Sapphire flight vehicle and engineering model. A user situated 30 miles from the ground station was able to remotely connect to the equipment, adjust radio parameters, and contact Sapphire. He ran a standard state-of-health check and logged out.

3.0 Conclusion and Future Work

Phase 3 was a successful year for the ASSET project. Several key team members were added and the conceptual underpinnings of the project were expanded. Most importantly, hardware and software elements of ASSET were introduced: the core operating system, electronic mail-based interfaces, a scheduler, a client interface, and internet-based remote control of ground systems. The Sapphire and Opal spacecraft were modified to incorporate beacon monitoring experiments through ASSET.

The main goals for ASSET in 1998 include:

(1) Begin controlled experimentation of beacon monitoring with Sapphire.
(2) Improve conceptual studies of health management and scheduling processes.
(3) Improve the health management and scheduling processes already implemented in ASSET.
(4) Fully integrate scheduling and product request into ASSET
(5) Expand internet-based remote control to include automated control.

4.0 References / Publications

The following papers were written in part to satisfy objectives of Phase 3 of the ASSET Project, and are referenced in this report.

✓[1] Kitts, Christopher A. and Michael A. Swartwout, "Experimental Initiatives in Space Systems Operations," *Proceedings of the Annual Satellite Command, Control and Network Management Conference*, Reston VA, September 3-5, 1997. Also presented at the 1998 INFORMS Conference, Monterey, CA, January 1998.

✓[2] Kitts, Christopher A., "The ASSET Interface: Balancing High Level Specification with Low Level Control," *Proceedings of the 1998 IEEE Aerospace Conference*, Snowmass, CO, March 21-28, 1998.

✓[3] Kitts, Christopher A., "Specifying Spacecraft Operations at the Product / Service Level," *Proceedings of the 48th International Astronautical Federation Congress*, Turin, Italy, October 6-10, 1997.

✓[4] Batra, Rajesh K., "The Design of a Highly Configurable, Reusable Operating System for Testbed Satellites," *Proceedings of the 1997 AIAA/USU Conference on Small Satellites*, Logan, UT, September 15-18, 1997.

✓[5] Swartwout, Michael A., and Christopher A. Kitts, "Automated Health Operations for the SAPPHIRE Spacecraft," *Proceedings of ITC/USA '96: The 33rd Annual International Telemetering Conference*, Las Vegas, NV, October 30-November 1, 1997.

✓[6] Swartwout, Michael A., "Engineering Data Summaries for Space Missions," *Proceedings of the 1998 IEEE Aerospace Conference*, Snowmass, CO, March 21-28, 1998.

✓[7] Swartwout, Michael A., Carlos G. Niederstrasser, Christopher A. Kitts, Rajesh K. Batra and Ken P. Koller, "Experiments in Automated Health Assessment and Notification for the Sapphire Microsatellite," *Proceedings of SpaceOps '98: The 5th International Symposium on Space Mission Operations and Ground Data Systems*, Tokyo, Japan, June 1-5, 1998.

Appendix

References /Publications

EXPERIMENTAL INITIATIVES IN SPACE SYSTEM OPERATIONS

Christopher A. Kitts* and Michael A. Swartwout**
Space Systems Development Laboratory
Stanford University
Stanford, CA 94305-4035

ABSTRACT

To develop and validate innovations for improving the cost-effectiveness of space operations systems, Stanford's Space Systems Development Laboratory is developing a comprehensive real-world spacecraft command and control system. This system will include a number of microsatellites, a number of geographically distributed ground communications stations, a central mission control complex, and an Internet and amateur radio communications network. The primary tasks of the system include the production of mission products and the maintenance of system health. Current research work is focusing on model-based techniques for anomaly management, a blackboard architecture for software control, a high level product specification interface, and a beacon system for efficient anomaly detection and notification. Overall, this testbed promotes rapid innovation and validation of high risk operational strategies. This paper reviews the experimental system and the nature of current research work.

INTRODUCTION

As a major lifecycle phase, operations can account for up to 60% of total program costs [1]. Poor integration of operational concerns into the design of missions and a reliance on experiential operational techniques habitually result in costly and human intensive mission control activities. This can limit the scope of current missions, deter the funding of new programs, and even discontinue missions with active spacecraft that are still technically capable of sustained service.

A number of recent space industry initiatives have placed significant competitive pressures on the manner in which space systems are operated. First, a variety of enterprises, both commercial and governmental, are beginning to deploy satellite constellations consisting of tens to hundreds of spacecraft in order to provide communications, navigation, and remote monitoring services. This development alone may lead to an order of magnitude increase in the number of operational spacecraft within the next decade; this poses a significant challenge given the limited scalability of current experiential operation techniques. Second, diminishing federal budgets are severely constraining the resources of civil and military space programs at a time when a vast array of end-user systems are beginning to rely upon these space-based elements. Dramatic improvements in operational approaches are required in order to exploit the significant benefits that these space-based systems have over their terrestrial counterparts. Third, the space industry is only now beginning to engineer systems with the technical capability and flexibility that enables the precise investigation of unknown remote domains. For example, NASA missions are being developed to investigate unknown environments in order to find and study "anything of interest". This vast abstraction of a typical mission statement requires fundamentally different approaches to system operations in which there is no experience upon which to rely and in which real-time human control is simply unachievable [2].

A variety of research programs have sought to address these issues by developing and introducing innovations in automation, reasoning approaches, architectural structure, decision-making location, design methodology, and a variety of other potential solutions. Such new techniques are typically validated through computer simulation, experimentation with a

* Doctoral Candidate, Mechanical Engineering, Stanford University. Caelum Research Corporation, NASA Ames Research Center, CA.
** Doctoral Candidate, Aeronautical and Astronautical Engineering, Stanford University.

ground-based testbed, and/or shadow operations within a real space system. The political, organizational, and risk related challenges that exist in most spacecraft mission architectures, however, often impede rapid implementation of these strategies into the day-to-day operations of space systems. A classic Catch-22 situation exists: innovative operational strategies will not revolutionize the space industry until they can be fully tested in a real-world environment, yet these real-world settings are typically too inflexible and risky to permit such an opportunity [3].

Stanford's Space Systems Development Laboratory (SSDL) is attempting to alleviate this dilemma through its Automated Space Systems Experimental Testbed (ASSET) research program. The ASSET system is a simple yet comprehensive real-world space operations network. This system will be used to operate a variety of academic and amateur microsatellites; in doing so, it will also serve as a low inertia, flexible, real-world validation testbed for new operational methods and technologies.

THE AUTOMATED SPACE SYSTEM EXPERIMENTAL TESTBED (ASSET)

Figure 1 shows a high level view of the ASSET mission architecture [3]. The basic components include the user interface, a control center, ground stations, communications links, and the target spacecraft. During the current developmental phase, a highly centralized operations strategy is being pursued with nearly all mission management decision making executed in the control center. These tasks include experimental specification, resource allocation throughout the ground and space segment, health management, contact planning, data formatting and distribution, and executive control.

Spacecraft. Three university microsatellites are currently being integrated into the ASSET system. SAPPHIRE, SSDL's first satellite, will characterize the space-based operation of experimental infrared sensors, photograph the Earth, and broadcast voice messages [4]. SAPPHIRE is currently undergoing final test; secondary launch options are being explored. Operations with Weber State University's WeberSat spacecraft, launched in 1990, will include Earth photography and telemetry analysis [5]. WeberSat's on-board software is currently being modified to accommodate these services. SSDL's second satellite, OPAL, will test a variety of inexpensive COTS sensors and will validate a launch mechanism for deploying hockey-puck sized science craft [6].

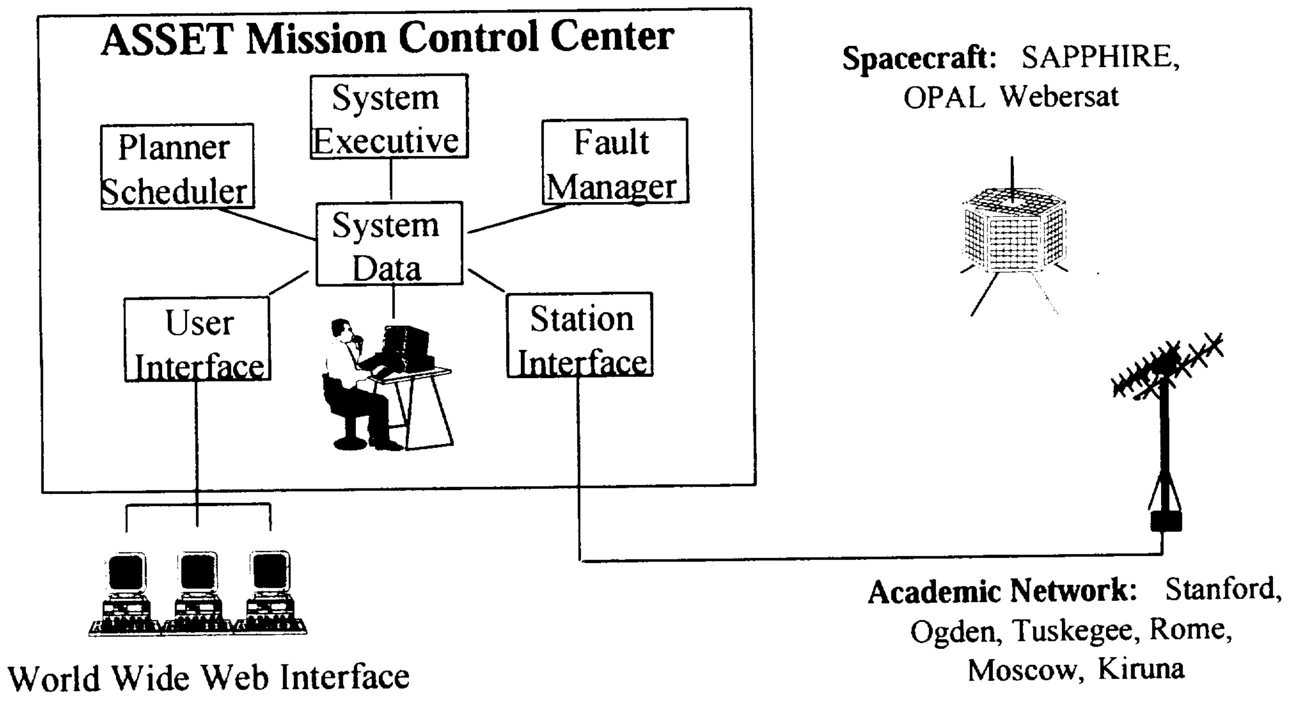

ASSET Mission Control Center
System Executive
Planner Scheduler
Fault Manager
System Data
User Interface
Station Interface
Spacecraft: SAPPHIRE, OPAL Webersat
Academic Network: Stanford, Ogden, Tuskegee, Rome, Moscow, Kiruna
World Wide Web Interface

Figure 1 - The ASSET Space System Architecture

In addition to these spacecraft, additional vehicles are being considered for future operations. These include all future SQUIRT series spacecraft built by SSDL [7], other university microsatellites currently under development, and some amateur spacecraft currently on-orbit. While these spacecraft have simple missions, it is worth noting that their mission products are reasonable operational analogs to the remote imaging, direct broadcasting, and sensor recording products offered by industrial, civil, and military space systems.

Groundstations. The ASSET groundstations employ HAM radio frequencies and equipment commonly used for amateur satellite communications. Typical stations have steerable antennae and use packet radio data formats. To date, facilities at Stanford and at Weber State University have been used for experimentation. Nearly a dozen other stations throughout the world (Sweden, Italy, Russia, Japan, and throughout the U.S.) have been identified for future integration. The resulting network will pose planning, scheduling, and execution challenges on par with those currently experienced within the Air Force Satellite Control Network (AFSCN), the NASA Deep Space Network (DSN), and a variety of other large scale space operations systems.

Mission Control Center. The ASSET mission control center resides at Stanford University and consists of several workstations and operators/developers. In the current centralized architecture, the agents in this control center are responsible for mission planning, resource scheduling, executive control, health management, and interfacing with external users and internal engineers. Much of SSDL's research is aimed at understanding these tasks well enough to support automated end-to-end mission products processing and systems health management.

In the developing mission architecture, clients submit requests for products through a World Wide Web interface; these requests are stored in a central database. Various software modules filter these and system-originated health monitoring requests in order to select products for processing, schedule the spacecraft and groundstation resources required, and plan the low level contact plans necessary. Contact plans are executed via the groundstations using the spacecraft-specific command and telemetry formats. Mission products are returned to the control center for delivery to customers and for storage in a searchable archive. Telemetry is analyzed in order to detect anomalies. Anomalous conditions trigger operator notification, rescheduling of resources to support contingency activities, and a variety of diagnosis and reconfiguration agents in order to assist spacecraft engineers.

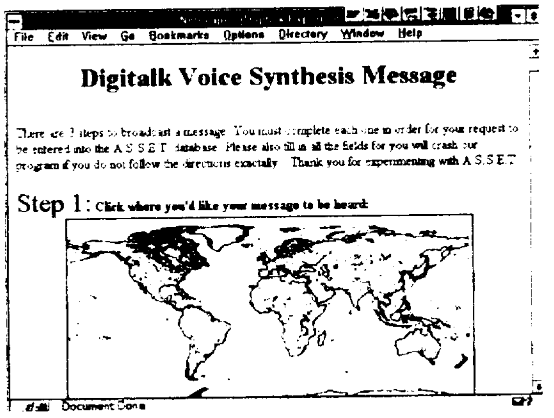
EXPERIMENTAL INITIATIVES

Current research projects within the ASSET program consist of developing innovations for both the processing of mission products as well as managing system health. Several of these initiatives are described here.

Product Level Client Interface. Requesting a product or service from a space system is typically an inefficient process [8]. First, requests are often directed to a human mission planner in a slow turn-around proposal process or in a manner that requires synchronization in time and/or space. Second, clients are often asked to specify their product at a level inappropriate to their knowledge or desire. For instance, clients only interested in obtaining the end product often must become involved with potentially complex and uninteresting details of the processing implementation, and vice versa. Third, the process used to specify a desired product commonly overconstrains the set of possible implementations; this can severely limit the flexibility of the mission planning process when resources are constrained.

The ASSET system addresses these problems by incorporating a simple World Wide Web based client interface capable of obtaining product requests at varying levels of abstraction. This interface decouples interaction with the mission control center so products may be specified at the convenience of the client. The interface leads the client through an interview process in order to gather information required to completely specify the product request. At the highest level, this process allows the client to provide only the pertinent attributes and delivery information concerning the product. At the lowest level, this process allows the client to specify specific implementation details or even the contact procedures themselves. Finally, the interface captures the client's product specification and submits it to the ASSET mission planning system. The specification is interpreted as the set of all implementations that can successfully produce the client's product. This set is only constrained by rational, committed decision-making aimed at generating an "optimal" schedule. One particularly novel technique in this regard is a software agent that examines the intersection of various product specification sets in order to intelligently select single product implementations that satisfy more than one client [9].

As an example of this style of interface, consider a request for a series of Earth photographs. The client selects the number of photographs and the interval between snapshots; other parameters such as resolution and filter setting are easily accommodated but are not options for the current ASSET spacecraft. To specify the subject of the photo at a high level, the client may select a geographic region by clicking on a world map; alternatively, a mobile object such as a person, vehicle, or weather system can be selected if this object has a forecasted itinerary registered with the ASSET system. Figure 2 shows a sample screen of the ASSET interface for a photograph product.



Digitalk Voice Synthesis Message

There are 3 steps to broadcast a message. You must complete each one in order for your request to be entered into the A.S.S.E.T. database. Please also fill in all the fields for you will crash our program if you do not follow the directions exactally. Thank you for experimenting with A.S.S.E.T.

Step 1: Click where you'd like your message to be heard:

Figure 2 - *The ASSET Client Interface*

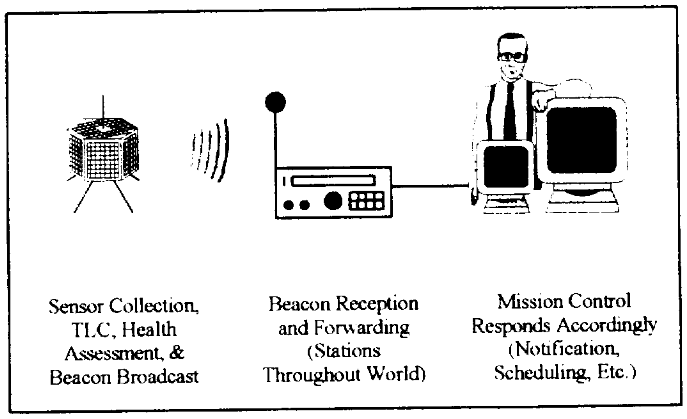

Sensor Collection, TLC, Health Assessment, & Beacon Broadcast
Beacon Reception and Forwarding (Stations Throughout World)
Mission Control Responds Accordingly (Notification, Scheduling, Etc.)

Figure 3 - *The SAPPHIRE/ASSET Beacon System*

Beacon-based Anomaly Notification. For many spacecraft, the vast majority of health analysis vehicle contacts result in a normal spacecraft health assessment. As an example, a recent Air Force study showed that this percentage was approximately 96% of contacts over a three year period [10]. This fact, combined with increasing confidence in the short-term effectiveness of telemetry limit checking (TLC) for anomaly detection, has inspired the concept of beacon operations.

In beacon operations, an automated TLC agent is combined with a low-cost communications link in order to reduce the cost, increase the response time, and generally maintain the detection capability of health management operations. Specifically, the TLC agent is situated on-board the spacecraft and passes a very limited amount of aggregate state data back to mission control through the communications link. The Air Force has considered a version of this concept for Earth orbiting vehicles with a dedicated network of receiving stations [11]. NASA is considering this concept for space probes [12].

Figure 3 shows version of beacon-based anomaly notification that is being incorporated within the ASSET system for the SAPPHIRE spacecraft [13]. SAPPHIRE has an on-board TLC system that supports commandable, state-dependent limits and aggregate telemetry checks. A two bit state of health signal is generated as a result of the TLC; the four various health states represent a conceptual mapping between vehicle health and required operator action. This health signal is periodically broadcast by SAPPHIRE's beacon system. A dedicated network of receive-only beacon receiving stations are being designed and deployed throughout the world by SSDL's academic collaborators. This network will detect SAPPHIRE's beacon broadcasts and forward the health information to ASSET's mission control center. At this point, the mission control center will initiate appropriate actions depending upon the state of health. For example, if health is normal, the message will simply be logged for future acknowledgment by the cognizant engineer. Alternatively, if SAPPHIRE is in an emergency state, the on-call operator will be notified, near-term product processing is put on hold, groundstation time for contingency operations is scheduled, and relevant engineering and operational documentation is recalled. In end-to-end validation of this technique, system level metrics such as cost, timeliness, and assessment quality will be compared to conventional performance measures.

Engineering Data Summary. In conventional human-based operations, satellite engineers spend hours closely watching the various spacecraft measurands; this enables them to spot trends and develop an intuitive feel for the various subsystems and the system's historical state. Such familiarity is extremely important for anomaly detection, diagnosis, and recovery since the effects of faults and control actions may not be completely observable by measurands.

With recent advances in automation, many missions have chosen or are considering the automation of various anomaly management tasks and their migration to the spacecraft. But in even the most advanced systems, human operators play a role. The challenge, then, is to maintain an adequate level of context for the operators even when they do not play a role in routine, day-to-day telemetry analysis. This is particularly difficult when historical telemetry is not available on the ground for simulated play-back due to the cost-cutting desire to reduce communications bandwidth.

An engineering data summary is a report, derived on-board during the filtering of real-time telemetry, that succinctly presents the most important spacecraft state information in order to generate the required operator context [14]. This summary can be periodically downloaded or can be requested during anomalous conditions in order to balance the need for operator awareness with the costs and time of raw telemetry transmission and analysis. A variety of strategies are being considered for generating the summary. These range from simply logging statistical information to only reporting deviations from simulated mission models. Various implementations are currently being studied: telemetry for NASA's Topex satellite is being processed for eventual shadow-mode verification; ground-based operational validation will be attempted within the ASSET system for a number of its spacecraft.

Model-Based Anomaly Management. Since its inception, the space community has relied almost exclusively upon experiential approaches to anomaly management. In this strategy, reasoning is based upon a collection of heuristics, intuitions, and past experiences. This style of knowledge base represents the fundamental design and behavior of the system in a very weak manner. In order to overcome the costs of this reasoning approach, especially when implemented with large numbers of highly trained human operators, model-based reasoning approaches are being investigated by a number of researchers. In this strategy, often referred to as reasoning from first principles, reasoning derives from a basic description of a system. Typical elements in a system description include a list of components, their connections, their functions, and their valid input/output values.

A solid theoretical foundation exists for fault detection and diagnosis. In this theory, a "fault" is defined as a deviation from expectation as predicted by the system description. Work within the ASSET program is seeking to extend this theoretical foundation to fault recovery, the management task that seeks to reconfigure the system in some manner to sustain the mission. Going beyond the identification of component redundancy, the resulting techniques are allowing functional redundancies to be identified even when the result is to use components in a non-traditional manner. In addition, a new type of anomaly, termed a "hazard", has been defined as a condition where a component is performing as expected but an unintended condition or behavior exists [2]. Expanding the theoretical types of anomalies is proving useful in relaxing the modeling requirements in the design phase.

Initially, these new techniques are being incorporated into the ASSET system as part of an engineering decision support system. This system will keep human operators in the loop during contingency operations and will assist in fault management operations by identifying anomalies, proposing troubleshooting procedures and candidate diagnoses, and suggesting courses of action. An important element of this job is to effectively communicate spacecraft state to the operator. In this regard, a simple and inexpensive Web interface is being developed to provide high level engineering analysis of spacecraft telemetry [8]. Using the Virtual Reality Modeling Language (VRML), three dimensional representations of the spacecraft, its environment, and its components can be generated and used for health management tasks. Figure 4 shows an orbital view of the SAPPHIRE spacecraft; in this particular display, the spacecraft's position, attitude, and lighting conditions are all graphically depicted. The operator can click on the spacecraft in order to be shown an internal view of the vehicle, as depicted in Figure 5. In this view, the state of the satellite can be represented in a variety of ways. For instance, the color of each box can be used to represent the temperature of the component. Other displays can show raw or filtered telemetry, graphical depictions of spacecraft status, and abstracted analysis diagrams; links to appropriate engineering manuals and operations documents are easily implemented. Compared to contemporary spacecraft displays, key features of this interface are the integration of information across subsystems, the use of models and causality as means of generating appropriate display features, and the abstraction of information compared to merely conveying raw data in novel ways.

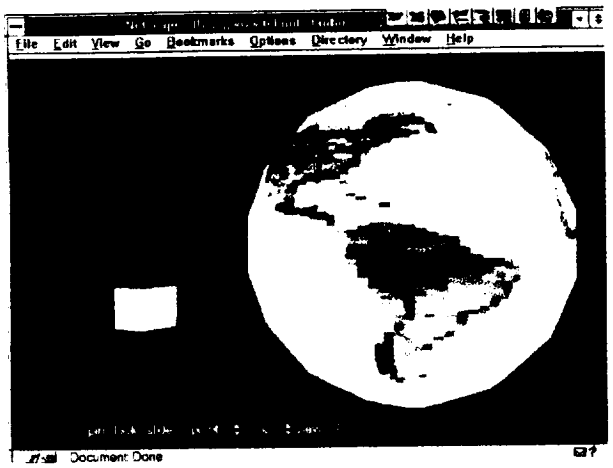


Figure 4 - VRML Display of SAPPHIRE in Orbit

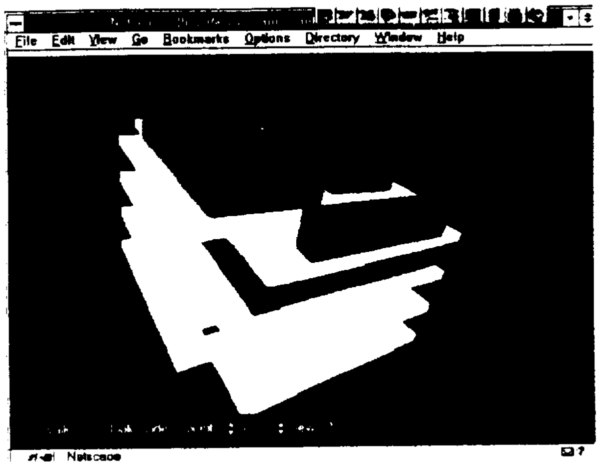


Figure 5 - VRML Display of SAPPHIRE's Interior.

Blackboard Based Software Control. For initial development, a blackboard architecture has been chosen as the software framework in which to manage tasks at the mission control center. A blackboard software control system consists of working memory (the blackboard), a collection of knowledge sources each of which contributes highly specialized inferencing capability, and a control strategy which dynamically invokes the appropriate knowledge source. The result is a collaborative problem-solving process in which all relevant data, observations, and decisions visibly pass through the blackboard in a controlled manner. Blackboard systems have been found useful in overcoming some of the design limitations of conventional knowledge based systems such as inflexible inference control and poor exploitation of partial problem solutions [10].

The selection of a blackboard architecture is of interest for several reasons. First, understanding how to adapt and implement it within a comprehensive space system is a worthy research application. Second, its potential value in supporting the migration of decision-making throughout the space system is intriguing. Third, its capability to flexibly integrate modular inference engines benefits the testbed nature of the project since a wide variety of decision-making techniques will eventually be tested.

Current implementation work for the ASSET blackboard system consists of operating two parallel blackboards, one for mission products management and one for system health management. Automated operator notification in response to a detected anomaly has already been demonstrated within the health management blackboard system.

Robust Groundstation Automation. The current ASSET amateur radio groundstations consist of inexpensive COTS equipment designed primarily for human operation. Development work is currently underway to expand automated control of these stations in order to configure equipment, collect station status information, and conduct command and telemetry operations with compatible spacecraft.

The system software being designed to support these functions is being specifically targeted to service direct human operation, human teleoperation, and script-based autonomous operation. The direct mode will enable on-site human operators to control the station in a manner consistent with current practice. Teleoperation mode will permit human operators to similarly control the station while located elsewhere. Autonomous mode will execute prepared scripts, composed by the ASSET system or human operators, in order to control the station. The scripting capability is being designed to support both time-based and event-based execution, conditional reasoning in support of contingency operations or opportunistic procedures, and the future decentralization of some overall space system functions. Special care is being taken to modularize this system so that the software is easily adaptable to other groundstation-specific equipment and procedures.

FUTURE WORK

Development of the ASSET system and its automated capabilities is an ongoing track of research within the SSDL. Work is progressing in all functional areas in order to achieve more advanced and cost-efficient capabilities. To ensure the applicability of this work, SSDL will continue to cooperate with both industry and governmental space organizations. These collaborations currently include beacon system validation, engineering data summarization, and client interface development. Increasing the level of collaboration with the AFSCN, Phillips Laboratory, and other military space agencies is of particular interest to SSDL.

Developmental progress is specifically being targeted to permit the expansion of the space system architecture so that it can accommodate more users, missions, experiments, spacecraft, and ground stations. In addition, increasing the level of autonomy, migrating control authority throughout the system, and applying new reasoning approaches are particular research areas that are being pursued. Finally, formally developing a suite of "Design for Operability" concurrent engineering methodologies is of special interest to the current set of researchers.

CONCLUSION

The ASSET system is proving to be a valuable prototype architecture for iteratively developing and validating operational innovations that will contribute to the performance and competitiveness of future space systems. As a research testbed, the development of the ASSET system provides rich fields of inquiry in the areas of user interfaces, planning and scheduling, health management, executive control, systems engineering, etc. The benefit to the spacecraft industry is clear: as a comprehensive, low inertia, flexible, real world validation testbed, the ASSET system will provide an unparalleled opportunity for experimentation with high risk operational technologies. Furthermore, the academic validation process will assist in supplanting anecdotal analysis commonly performed within the space community with standard evaluation practices aimed at assessing overall system competitiveness.

Several elements of the current work are already contributing directly to NASA's New Millennium Program and Deep Space Network. These initiatives provide SSDL students with exciting engineering problems; collaborators, in turn, receive fresh, experimentally tested innovations in operational strategies. It is SSDL's hope that this unique research option will significantly accelerate the development of more cost-effective end-to-end space system operations.

ACKNOWLEDGMENTS

The authors wish to thank current and former research students who have contributed to the development of the ASSET operations network. In addition, special thanks is extended to Professor Robert Twiggs, the SSDL Director, for his support and guidance in the administration of this research program. The NASA Jet Propulsion Laboratory and Ames Research Center are acknowledged for their support and assistance with this research project. Finally, appreciation is extended to all of the institutions and organizations that have participated in, contributed to, and provided feedback concerning various aspects of the ASSET system. This work has been performed in partial satisfaction of graduate studies at Stanford University.

REFERENCES

[1] Ely, Neal, and Thomas P. O'Brien, "Space Logistics and Reliability", in Space Mission Analysis and Design, ed. James R. Wertz and Wiley J. Larson, 633-656. London: Kluwer Academic Publishers, 1991.

[2] Kitts, Christopher A., Theory and Experiments in Model-Based Space System Operations, Draft Research Report, Space Systems Development Laboratory, Stanford University, 1997.

[3] Kitts, Christopher A., "A Global Spacecraft Control Network for Spacecraft Autonomy Research." Proceedings of SpaceOps '96: The Fourth International Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, September 16-20, 1996.

[4] Kitts, Christopher A., and William H. Kim, "The Design and Construction of the Stanford Audio Phonic Photographic Infrared Experiment (SAPPHIRE) Satellite", Proceedings of the 8th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, Aug. 29 - Sept. 1, 1994.

[5] Bonsall, Charles, ed., WeberSat Users Handbook, Edition 1.0, Center for Aerospace Technology, Weber State University, January 1991.

[6] Engberg, Brian, Jeff Ota, and Jason Suchman, "The OPAL Satellite Project: Continuing the Next Generation Small Satellite Development", Proceedings of the 9th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 19-22, 1995.

[7] Kitts, Christopher A., and Robert J. Twiggs, "The Satellite Quick Research Testbed (SQUIRT) Program," Proceedings of the 8th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 16-22, 1994.

[8] Kitts, Christopher A., and Clemens Tillier, "A World Wide Web Interface for Automated Spacecraft Operation", ITC/USA '96: Proceedings of 32nd Annual International Telemetering Conference, October 28-31, 1996.

[9] Kitts, Christopher A., "Specifying Spacecraft Operations at the Product/Service Level", Pending publication in the Proceedings of the 48th International Astronautical Federation Congress, Turin, Italy, October 6-10, 1997.

[10] Farmer, Mike and Randy Culver, "The Challenges of Low-Cost, Automated Satellite Operations," Proceedings of the 31st International Telemetering Conference, Las Vegas, Nevada, October 30-November 2, 1995, pp. 203-209.

[11] Hovanessian, S. A., S. H. Raghavan, and D. A. Taggart, LIFELINE: A Concept for Automated Satellite Supervision, Aerospace Report No. TOR-93(3516)-1, August 1993.

[12] Wyatt, E. Jay, and John B. Carraway, "Beacon Monitoring Approach to Spacecraft Operations", Reducing the Cost of Spacecraft Ground Systems and Operations, Oxfordshire, United Kingdom, September 27-29, 1995.

[13] Swartwout, Michael A., and Christopher A. Kitts, "A Beacon Monitoring System for Automated Fault Management Operations", Proceedings of the Tenth Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 16-19, 1996.

[14] Swartwout, Michael A., and Christopher A. Kitts, "Automated Health Operations for the SAPPHIRE Spacecraft", Pending publication in the ITC/USA '97: Proceedings of the 33rd Annual International Telemetering Conference, Las Vegas, Nevada, October 30 – November 1, 1997.

[15] Smaill, Alan, Artificial Intelligence: Knowledge Representation and Inference, Teaching Note, Edinburgh University, December 1995.

The ASSET Client Interface: Balancing High Level Specification with Low Level Control

Christopher A. Kitts *
Space Systems Development Laboratory
453 Durand Building
Stanford University, Stanford, CA 94305-4035
(650) 725-6794
kitts@leland.stanford.edu

Abstract - This paper describes an advanced client interface for spacecraft operations. This interface permits system clients to specify operations at a level of abstraction relevant to their experience, need, and desire. The interface encourages high-level specification of desired products in which only the relevant product attributes are described. This mode is simple, convenient, and promotes system level flexibility through the elimination of unnecessary constraints. More sophisticated clients may augment this process with lower level directions that mandate particular command and telemetry operations or constrain specific operational variables.

The design of this interface is guided by a Product Specification Model which relates the attributes of general space-based products to the underlying command and telemetry operations that generate those products. By exposing all of the Model's properties to the interface's interview process, a varying level of abstraction is supported during product specification.

This interface is being incorporated into a real-world, experimental mission operations system that consists of several amateur and university-built microsatellites, a global network of remote groundstations, an Internet and amateur radio-based communications infrastructure, and a central mission control center. Initial experimentation suggests that this style of specification capability will contribute to the effectiveness of space systems by 1) enhancing ease of use while conserving the level of control required by some clients, 2) providing a framework for automating the generation of product production procedures, 3) eliminating costly overconstraints commonly imposed through lower level specification, and 4) identifying opportunities to generate a single product that satisfies multiple customers.

This paper describes the operational issues of high and low level specification and presents the conceptual framework for the developed Product Specification Model. The design of the resulting advanced user interface and its implementation within SSDL's mission operations system is also described.

TABLE OF CONTENTS

1. INTRODUCTION

Declining federal budgets and an array of commercial initiatives are providing a significant impetus to improve the competitiveness of space systems. This involves lowering the cost, reducing the cycle time, and increasing the quality and features of a system's products and services. These innovations are particularly important for the broad class of space systems that produce discrete, custom, and/or on-demand products. Such systems include a large segment of NASA science missions, military and commercial space-based imaging systems, as well as certain broadcasting services. NASA's "Science from a Laptop" initiative is one example of a technology program targeted at improving the competitiveness of such services.

As part of its research program, Stanford University's Space Systems Development Laboratory (SSDL) is also developing innovations in this domain. These advancements are being validated through the use of the Automated Space System Experimental Testbed (ASSET). ASSET is a real-world mission operations system which consists of several amateur and university-built microsatellites, a global network of remote groundstations, an Internet and amateur radio-based communications infrastructure, and a central mission control center [1].

In improving space system competitiveness, a particular technological focus has been to improve the process by which clients request products or services. In many

* Doctoral Candidate, Stanford University.

contemporary space systems, this is typically an inefficient process. First, the mechanism by which the request is represented, such as a proposal or form, is often difficult and tedious to use. Second, submission can be slow due to the mode of delivery and the need for iterative, synchronized consultation with a human mission planner. Third, clients are often exposed to complex processing details, such as equipment configurations or station visibilities, that are inappropriate to their level of knowledge or interest. Fourth, the process often forces or induces an overconstrained specification that is actually a small subset of the possible operational implementations that could effectively satisfy the client.

Early SSDL work that aimed at solving these inefficiencies involved the development of a Web-based client interface. This interface collected all necessary specification parameters as part of a context-sensitive interview process [2]. This system directly addressed problems of speed, synchronization, and convenience by automating the request process through a widely accessible, common, and graphical client-server scheme.

An extension to this laid the groundwork for high-level product specification by modifying the interview process to collect product attributes [3]. During this phase, the incorporation of JavaScript significantly improved the performance of the automated interface; JavaScript allowed the migration of request processing to the client's terminal thereby reducing interview delays due to server and communications load.

Current work in this area focuses on 1) generalizing the high-level view of a discrete space product and 2) formalizing the model that relates this view to a set of consistent low-level plans for generating the product. With such a foundation, the resulting client interface is being implemented to exploit this product model by allowing users to specify operations through a combination of high and low-level constraints. The resulting system combines the features of convenient high-level direction with the full authority of low-level control. The interface is being integrated into the ASSET system in order to 1) take advantage of the flexibility afforded by a complete specification of all possible operational options and 2) identify common operational implementations that can satisfy more than one client. As an added feature to further improve the performance of the Web-based interview process, Dynamic HyperText Markup Language (DHTML) is being combined with JavaScript to significantly speed processing and reduce download times.

In describing the design and implementation of the ASSET client interface, the ASSET system if first reviewed. Next, the operational issues of high and low level specification are presented, and the developed Product Specification Model is described. A descriptive tour of selected pages of the implemented client interface is then presented. Finally, future work planned for the interface is noted and general conclusions are drawn.

2. THE ASSET SYSTEM

One of SSDL's primary research activities is the development and validation of new operational innovations that contribute to the system level competitiveness of space systems. In order to conduct this work in a relevant experimental environment, ASSET system is being developed. The ASSET system is a simple yet comprehensive real-world space operations network. This system will be used to operate a variety of academic and amateur microsatellites; in doing so, it will also serve as a low inertia, flexible, real-world validation testbed for new operational methods and technologies.

Figure 1 shows a high level view of the ASSET mission architecture [1]. The basic components include the user interface, a mission control center, groundstations, communications links, and the target spacecraft. During the current developmental phase, a highly centralized operations strategy is being pursued with nearly all mission management executed in the mission control center. These tasks include product specification, resource allocation throughout the ground and space segment, anomaly management, contact planning, data formatting and distribution, and executive control.

Spacecraft

Four university microsatellites are currently being integrated into the ASSET system. SAPPHIRE, SSDL's first satellite, will characterize the space-based operation of experimental infrared sensors, photograph the Earth, and broadcast voice messages [4]. SAPPHIRE is currently undergoing final testing; secondary launch options are being explored. Operations with Weber State University's WeberSat spacecraft, launched in 1990, will include Earth photography and telemetry analysis [5]. WeberSat's on-board software is currently being modified to accommodate these services; during the past year, however, WeberSat has been experiencing intermittent CPU resets which have hampered development. SSDL's second satellite, OPAL, will test a variety of inexpensive commercial off-the-shelf sensors and will validate a launch mechanism for deploying hockey-puck sized science craft [6]. The Barnacle microsatellite, a joint mission between SSDL and Santa Clara University, is being designed to characterize experimental fluxgate magnetometers and a low-cost spacecraft processing system [7]. While these spacecraft have simple missions, it is worth noting that their mission products are reasonable operational analogs to the remote imaging, direct broadcasting, and sensor recording products offered by many industrial, civil, and military space systems.

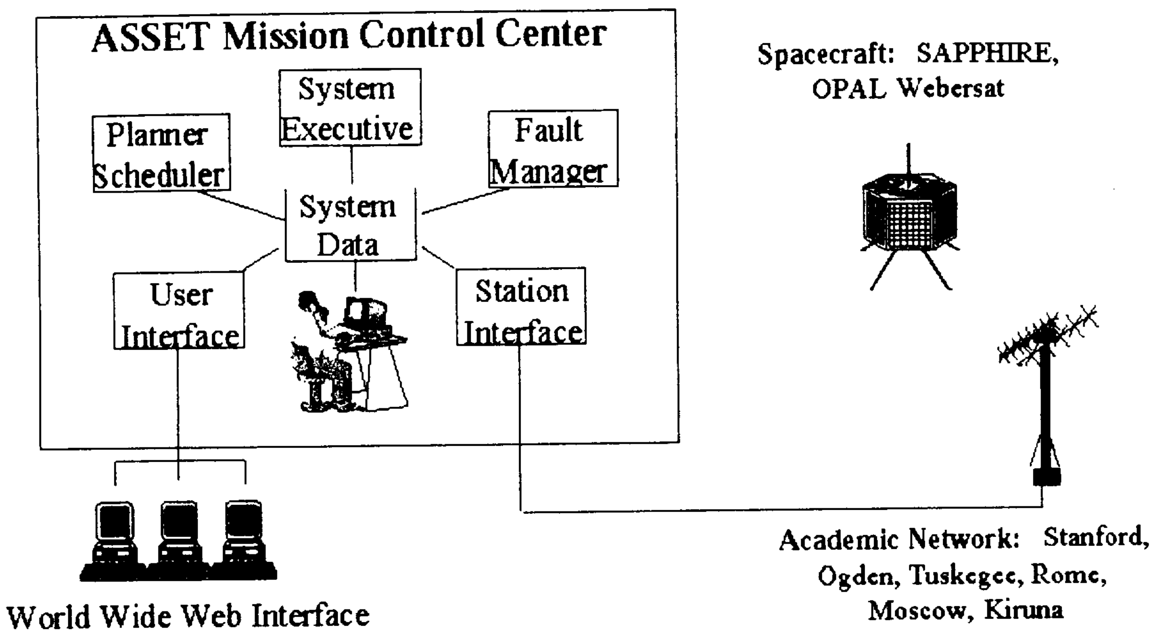

ASSET Mission Control Center
System Executive
Planner Scheduler
Fault Manager
System Data
User Interface
Station Interface
Spacecraft: SAPPHIRE, OPAL Webersat
World Wide Web Interface
Academic Network: Stanford, Ogden, Tuskegee, Rome, Moscow, Kiruna

Figure 1 - The ASSET Space System Architecture

Groundstations

The ASSET groundstations employ HAM radio frequencies and equipment commonly used for amateur satellite communications. Typical stations have steerable antennae and use packet radio data formats. To date, facilities at Stanford and at Weber State University have been used for experimentation; the Stanford station runs software capable of supporting both tele-operation and automated, bent-pipe, programmable control. Nearly a dozen other stations throughout the world (Sweden, Italy, Russia, Japan, Saudi Arabia, and across the U.S.) have been identified for future integration. The resulting network will pose planning, scheduling, and execution challenges similar to those currently experienced within the Air Force Satellite Control Network (AFSCN), the NASA Deep Space Network (DSN), and a variety of other large scale space operations systems.

Mission Control Center

The ASSET mission control center resides at Stanford University and consists of several workstations and operators/developers. In the current centralized architecture, the agents in this control center are responsible for mission planning, resource scheduling, executive control, health management, and interfacing with external users and internal engineers. SSDL's research is aimed at understanding these tasks well enough to support automated end-to-end mission products processing and system health management.

System Processing

In the developing mission architecture, clients submit requests for products through a World Wide Web interface; these requests are stored in a central database. Various software modules filter these and system-originated health monitoring requests in order to select products for processing, schedule the spacecraft and groundstation resources required, and plan the low level contact plans necessary. Contact plans are executed via the groundstations using the spacecraft-specific command and telemetry formats. Mission products are returned to the control center for delivery to customers and for storage in a searchable archive. Telemetry is analyzed in order to detect anomalies. Anomalous conditions trigger operator notification, rescheduling of resources to support contingency activities, and a variety of diagnosis and reconfiguration agents in order to assist spacecraft engineers.

System Products and Services

Three high level product/service offerings are currently being developed for external clients within the ASSET framework. Earth photography can be provided by both SAPPHIRE and WeberSat. Synthesized voice broadcasting is generated by SAPPHIRE. Data collection services are provided by all four spacecraft. Data collection is being used specifically for sensor characterization on SAPPHIRE, OPAL, and Barnacle; it is being used for health management operations for SAPPHIRE, OPAL, and WeberSat.

3. HIGH LEVEL DIRECTION VS. LOW LEVEL CONTROL

Before developing the relationships between high and low-level product specification, it is worth considering associated operational issues. These include the benefits obtained through high level direction, the barriers to obtaining true high level specifications in conventional space systems, and why low level specification is still necessary [8].

Benefits of High Level Direction

Simplicity is perhaps the most obvious benefit of being able to direct the actions of a system at a high level. Because an informed user generally knows what is to be achieved, merely describing this end result, typically a product or service, requires no other knowledge concerning the particular structure or behavior of the system. For example, a photograph may be ordered merely by specifying the subject of the photo, the deadline for delivery, and any relevant photographic parameters such as resolution, light level, etc. No knowledge concerning the capabilities or orbits of specific spacecraft, the location of communications stations, the functional status of the system, the system's command and telemetry procedures, or the nature of other product requests are required in order for the client to describe the product of interest. All operational concerns of this nature are transparent to the customer.

An additional benefit of high level direction is that it often results in a less constraining specification for system actions. At the lowest level of control, a request for system action is equivalent to specifying a single possible implementation; this is a fully constrained request in which no operational variable is left as a degree of freedom. Often, however, there are many possible ways in which a system may generate a product with particular attributes. Requesting the product at this high level is equivalent to implicitly specifying a set of many possible implementations; each of these implementations can adequately produce the desired product.

For example, a low level request for a photo to be taken at a particular time results in a single implementation for obtaining the picture of a desired object such as North America. Alternatively, a high level request for the same picture, specified simply as being of North America, may be satisfied by taking the photo any time when North America is in view. More precisely, the operational variable of time is limited to a single value for the low level request. It is constrained only to a time range or a series of time ranges for the high level request; these time ranges are derived through knowledge of the spacecraft's orbit, attitude, field of view, and other system parameters.

Capturing a broader set of possible implementations for a particular request increases the overall system flexibility for satisfying that request. The resulting degrees of freedom can be exploited to accommodate other product requests and/or to optimize product generation with respect to resource utilization.

For these reasons, the ASSET client interface is being designed to accept high level specifications. Furthermore, the planning, scheduling, and execution elements of the ASSET product management system are being integrated in order to exploit the resulting benefits.

Obstacles to Capturing True High Level Directives

Although high level direction capability provides a number of benefits, its integration into large scale space systems is often hampered.

One cause of this is that clients often overconstrain their true goals with particular suggestions on how to achieve that goal. For example, a client may ask for a photograph taken by a specific spacecraft when a variety of spacecraft may be suitable for obtaining the desired photo. This occurs due to attempts to be helpful; the client knows that such a constraint is satisfiable. It also occurs due to ignorance; the client may be unaware of alternative system capabilities that may be more desirable when the client's request is balanced with other system commitments.

Even when the client offers a true high level directive, the process that captures this specification may not be designed to represent requests at such a high level. This can be due to an inflexible method for representing and processing request data. It can also occur due to inefficient experiential processing of the request that effectively compiles it at a lower level. For example, a general request for a photo might be recorded by a mission planner as a request for a photo by a specific spacecraft simply because the planner knows that other spacecraft typically experience heavy operational loads. If this is done to simplify the capture process rather than as a controlled system planning heuristic, then potential operational implementations are needlessly ignored.

Another barrier is the aversion to enlarging the planning and scheduling search space; this certainly occurs if a full set of operational implementations is captured for each system request. But this search space can be easily decomposed based upon heuristics relevant to the planning process rather than those optimized to simplify the capture process. With this approach, the pruning becomes a rational element of the overall system rather than existing as an artifact of limited comprehension and poorly designed functional and/or organizational interfaces.

Finally, specifications are often inappropriately influenced by the request-time state of the space system. For instance, possible product implementations may be ignored if they require resources that are projected to be unavailable. But resource availability is often dynamic due to modifications of system tasking and to the evolving status of resource supplies. As a result, small changes in the system's state can radically alter the available implementations for a particular product; the sacrificed implementations may then become missed opportunities.

To address these barriers, the ASSET interface encourages the specification of only high level product attributes unless lower level control is necessary. A fundamental system model is used to generate a set of low level implementations consistent with the high level specification. This transformation is done without regard for the request-time state of the system. In this manner, operational flexibility is conserved through the specification interface.

The Continuing Need for Low Level Control

Even with the benefits of high level direction, there are still a number of reasons why low level specification is still a desirable feature. First, using a low level format and language for a request may simply be preferred by the client. This can be especially true for principle investigators who have an intimate knowledge of a scientific payload. It can be useful, however, to make the client aware of any additional costs incurred by this preferred manner of product specification.

Another reason for justifying low level control capability is that the high level specification process may fail to permit control of relevant processing parameters. This certainly occurs with poorly designed interfaces. It also occurs when basic assumptions have been made to provide a simple interface to the majority of system clients. This situation may also develop when the user community develops interest in controlling an attribute of the product not originally assessed as relevant. Furthermore, the user community may discover entirely new applications consistent with the capabilities of the system but completely distinct, at a high level, from the original product offerings. In each of these cases, the high level needs for specification will lag the high level capture capability of the interface; this can be extreme with large scale, complex, and high inertia mission control organizations. Permitting low level control ensures that the system is still applicable to the needs of the clients.

Finally, multi-product considerations, which could be accommodated by permitting direction at a level even higher than the current product level, can require the need for low level control. For example, a campaign of photographs or the gathering of a collection of products that provide any sensory information on an object are legitimate goals for clients. In the absence of campaign level specification, low level control authority provides a means by which a series of product requests can be tailored to the needs of a client.

Overall, the ASSET system addresses these issues by incorporating low level control options into its client interface.

4. The Product Specification Model

In this analysis, a service is defined as a general capability to provide products to clients. A product is defined as a specific action, a tangible artifact, or a set of information that provides value to a client. In order to generate products, particular system configurations are required. The specification of a product is therefore used to constrain the system's configuration at product generation time.

An improper mapping from product to system constraints can result in either of two situations. An underconstrained system permits the generation of inadequate products. An overconstrained system eliminates product implementations that may be optimal from the system's perspective. The methodology by which the specification to system constraint mapping is made therefore has a significant effect on the overall system performance. By developing and applying fundamental models relating the relevant product and operational parameters, a systematic framework can be implemented. This should support improved system efficiency and automation.

In presenting the model currently being developed for implementation in the ASSET system, this section first provides a simplified view of typical system elements. The manner in which a product specification constrains the configuration of these elements is then described. The Product Specification Model used to capture a client's request for a product is reviewed. Finally, observations concerning this modeling approach are made.

The Basic System Schema

Figure 2 shows a simplified conceptual schema for a product processing system of interest. In modeling this schema, the Object-Role Modeling (ORM) technique has been adopted [9]. Previous work on this project used the Entity-Relationship technique for modeling the same domain [10]. The ASSET researchers have found the ORM technique to be better suited to conceptualizing the space processing system primarily due to the manner in which ORM represents objectified relationships.

In the schema of the ASSET domain, system entities capable of generating products are generally classified as tools. These tools have a variety of possible behaviors/capabilities as well as an array of configuration parameters. Some of these configuration parameters are directly controllable. An example of this type of system is the SAPPHIRE voice broadcasting subsystem which has the capability of synthesizing and transmitting voice messages. The subsystem's directly controllable parameters include the message's contents, the message's repeat parameters (number of repeats and interval between repeats), the time of broadcast, and the transmission power. Its constant configuration parameters include transmission frequency, modulation parameters, antenna gain, and orbital elements.

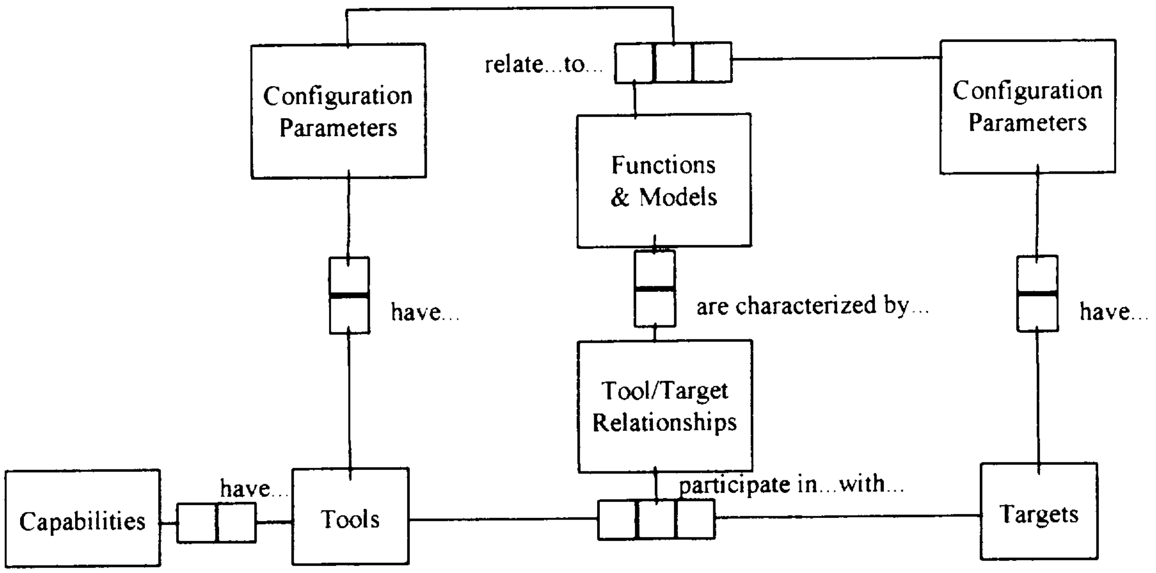


Figure 2 - A Simple System Schema

System entities that are the subject of the system's products or services are generally classified as targets. These targets also have a variety of configuration parameters most of which are not directly controllable in the short term. As an example, a person with a radio receiver would be a relevant target for SAPPHIRE's broadcast system. Configuration parameters for this target include location, receive frequency and performance (noise, sensitivity, etc.), antenna pointing and gain, and environmental conditions (such as rain, etc.).

Tools and their targets participate in roles as part of general relationships. These relationships are characterized by logical and/or mathematical functions or models. The variables in these models relate the tool and target configuration parameters. With respect to the SAPPHIRE voice broadcasting example, there is a line of sight existence/nonexistence relationship between the tool and target entities; this relationship is a function of the locations of the tool and the target. Other relevant relationships include the compatibility/incompatibility of communications link parameters and the existence/nonexistence of adequate link margin.

It should be noted that parameters and relationships relevant to other products also exist. These include items such as light level, orientation, distance, etc.

Specifying a Product

The act of specifying the generation of a product constrains the choice of tools and targets, their parameters, and their relationships. The specification can be made at a low level in which precise tool requirements are levied, or it can be made at a high level in which the attributes of the resulting product are defined.

For the voice broadcasting example, a low level specification would dictate values or ranges for the following variables: tool choice (the SAPPHIRE voice broadcasting subsystem), transmit power, broadcast time, message contents, and message repeat parameters. On the other hand, a high level product attribute specification would constrain values for the following variables: product type (a voice synthesized broadcast), intended recipient (i.e. target), deadline for receipt, message contents, and message repeat parameters.

In both cases, the constrained variables are elements of the system schema pictured in Figure 2. In the low level case, the processing parameters are specified, and a product results from these controlled actions. In the high level case, the product is specified, and a consistent set of processing parameters are derived in order to control the system.

The transformation from high level product attributes to low level processing parameters is performed by 1) directly constraining low level processing parameters when they directly correspond to high level product attributes, 2) constraining the values of the tool/target relationships, and 3) using the constrained tool/target relationships to calculate additional low level processing constraints from the remaining high level attributes.

For example, a typical high level specification includes a particular product type. This product type will require a particular type of processing capability which, in turn, limits the choice of tool. In addition, product type will constrain certain tool/target relationships. To continue the previous example, specifying a voice broadcast as the product type 1) limits tools to those capable of synthesizing and broadcasting voice, 2) requires that a line of sight exists between the tool and the target, 3) mandates communications compatibility between the tool and target, and 4) requires an adequate link margin for the broadcast.

For high level product specification, the product schema and client-supplied product attributes constitute the knowledge base from which consistent system configurations may be derived. These configurations are implicitly defined by the intersection of derived constraints upon the controllable tool configuration parameters. Explicit members of this set characterize the range of low level command parameters required for suitable product generation.

A Closer Look at Constrained Tool/Target Relationships

The process of deriving low level processing parameters from high level product attributes ranges from simple to complex. For example, the line of sight requirement for the voice broadcasting product decomposes into a simple constraint on possible processing times based upon the target's location and SAPPHIRE's orbital elements. This is largely due to the simplicity of the SAPPHIRE microsatellite which has no thrust capability.

On the other hand, a fully specified voice broadcasting product also requires adequate link quality. This results in a far more complex relationship as is shown in Figure 3. The physical constraints are primarily based upon standard communications link models [11]. The tool constraint refers to SAPPHIRE-specific design relationships [12]. Finally, the product constraint mandates adequate link quality; this essentially requires that the broadcast's signal to noise ratio is adequate for reception given the receiver sensitivity.

Reasoning proceeds along the following path. The product constraint mandates a certain minimum value for the broadcast's signal to noise ratio. A physical communications constraint transforms this into a minimum receiver to noise power ratio (since SAPPHIRE's broadcast modulation parameters are fixed). This effect propagates throughout the parameter space due to the enforced constraints. The overall result is that the original product constraint results in coupled limitations on two low level processing parameters: the transmission power and the broadcast time.

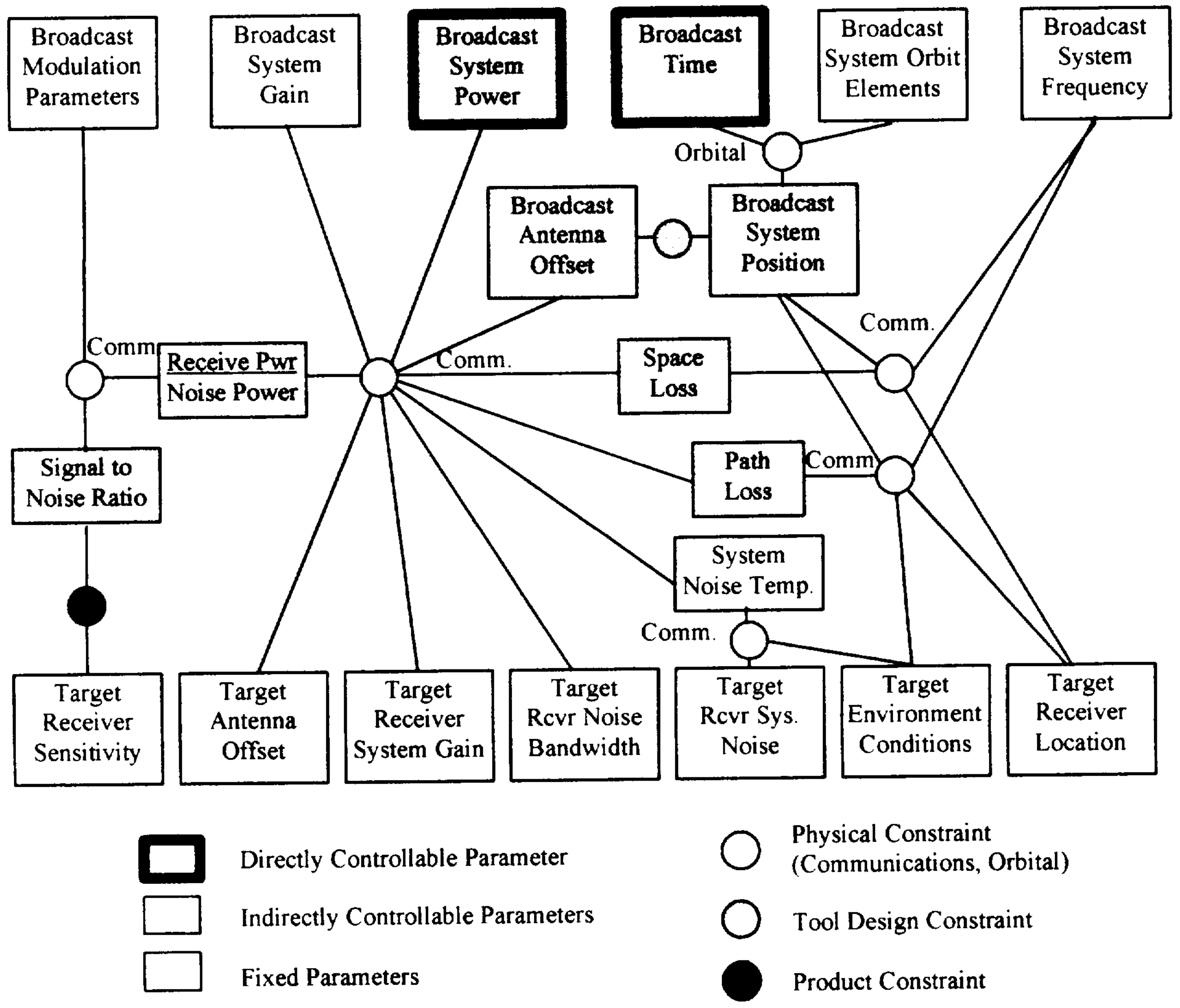

Broadcast Modulation Parameters
Broadcast System Gain
Broadcast System Power
Broadcast Time
Broadcast System Orbit Elements
Broadcast System Frequency
Orbital
Broadcast Antenna Offset
Broadcast System Position
Comm.
Comm
Receive Pwr
Noise Power
Comm.
Space Loss
Path Loss
Comm
Signal to Noise Ratio
System Noise Temp.
Comm.
Target Receiver Sensitivity
Target Antenna Offset
Target Receiver System Gain
Target Rcvr Noise Bandwidth
Target Rcvr Sys. Noise
Target Environment Conditions
Target Receiver Location
Directly Controllable Parameter
Indirectly Controllable Parameters
Fixed Parameters
Physical Constraint (Communications, Orbital)
Tool Design Constraint
Product Constraint

Figure 3 - Link Constraints for the Voice Broadcasting Product

The Product Specification Model

The ASSET interface adapts to the needs of the client by accepting product specifications at a high level, a low level, or through a mix of the two. This is accomplished by dynamically creating and populating a product request object during the interview process. The properties of this product request object include all relevant parameters from the system schema in which there is an operational degree of freedom. By exposing all of these properties to the request process, a varying level of abstraction is supported during product specification.

The Product Specification Model serves as the template for this process by defining the hierarchy of the product request object, the context-sensitive properties, and the constraints between properties. Figure 4 shows a simplified version of the model's object hierarchy. The first generation of properties includes information corresponding to request-specific data, product attributes, and product generation processing parameters. Subsequent generations in the hierarchy further characterize the requested specification; object properties may themselves be objects.

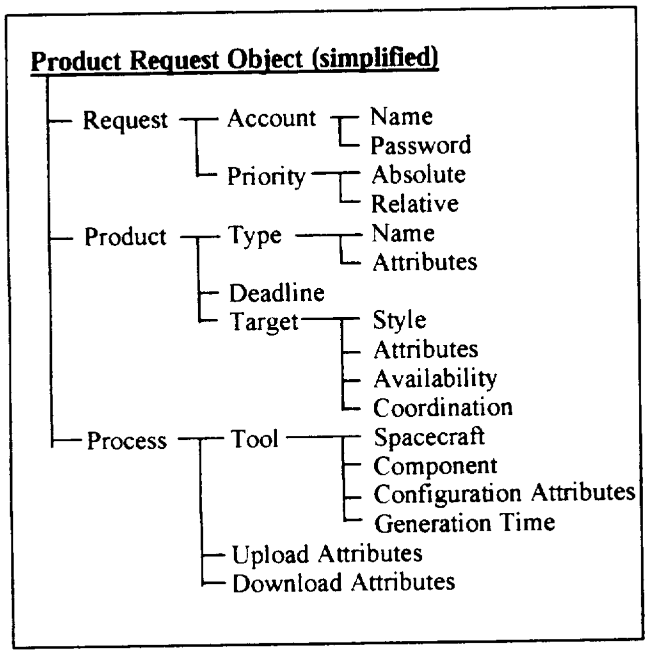


Figure 4 - A Simplified Product Specification Model

Some properties are defined for all possible specifications. Examples of this are the properties in the "Request" portion of the hierarchy; all requests will be assigned these specific properties, either by client specification or by derivation, since they are required for subsequent product processing. On the other hand, some properties, labeled "attributes" in Figure 4, are generic place holders; the object structure of these properties is defined during the interview process based upon previous client input. For instance, the "Product-Type-Attributes" property is an object with varying structure based upon the value of the "Product-Type-Name" property. If a client requests a voice broadcast product, then "voice broadcast" is assigned to "Product-Type-Name" and the internal structure of "Product-Type-Attributes" is defined by the properties "message content", "repeat number", and "repeat interval". A different set of "Product-Type-Attributes" properties are instantiated if a different product type, such as a photograph or data collection, is specified. The process of dynamically creating the structure of the product request object ensures that only relevant properties exist within the specification.

Many of the properties within the product request object are related due to the previously explained tool/target relationships. As a simple example, the specification of a product type will constrain the generation tool to those with the necessary capabilities. Similarly, specifying a specific target for a voice broadcast will place a constraint on processing time since a line of sight must exist between the tool and target.

Validation of the product request is accomplished through the use of both client and server processing. This process ensures a consistent specification with at least one existing implementation. When validated, the specification implicitly represents the set of all possible operational implementations for satisfying the client. Subsequent product processing, such as balancing the needs of competing requests, will add additional constraints to the original specification in order to eventually arrive at the specific set of procedures to be implemented.

Observations

By observing the functions and constraints relating high level product attributes and low level command and telemetry procedures, three particular characteristics are apparent.

The first is a one-to-many product-to-implementation quality. This confirms the general statements asserted earlier concerning the benefits of high level specification in giving a broad set of options. The resulting increase in system flexibility can be exploited by optimizing product generation and/or by increasing system throughput.

A more subtle characteristic is a potential many-to-one product-to-implementation quality. This highlights the fact that multiple product requests may be satisfied by a single operational implementation. For example, the same photograph of a region on the earth might have value for a resident of that region, a scientist gathering photographs of the entire earth, and a spacecraft engineer wishing to test and calibrate the camera. The ASSET planning system is being designed to identify and take advantage of opportunities of this nature. This is being accomplished by implementing a planning/scheduling preprocessor that identifies intersections between the implementation sets of different request specifications.

Finally, it should be pointed out that many services can be decomposed into a series of elemental products. While much of the previous discussion has focused on the system configuration at the time of product generation, post-generation operations are often required in order to retrieve products. For example, with SAPPHIRE, once a photograph has been generated (i.e. taken and stored into spacecraft memory), it must be retrieved from the spacecraft for subsequent delivery. This can be interpreted as a low-level data transfer product that is used to support delivery of client-level products. The ability of the product specification schema to adequately characterize product representations at a variety of levels such as this attests to the generality and value of the developed framework.

5. Implementation in the ASSET System

The principles and models described in this paper are being used to design and implement the client interface for the ASSET space system. This section provides a brief tour of the client interface.

The client interface consists of an automated, context-sensitive interview process that collects relevant specification parameters and submits these to the ASSET database. The interface is a series of Web pages written in HTML, DHTML and JavaScript. Upon request, the ASSET Web server relays the proper files to the client's Web browser. By supporting client-side processing and real-time page composition and layout, the use of DHTML and JavaScript permits speedy contextual processing during the interview, support for a variety of advanced interface elements, and simple state management of the interview process.

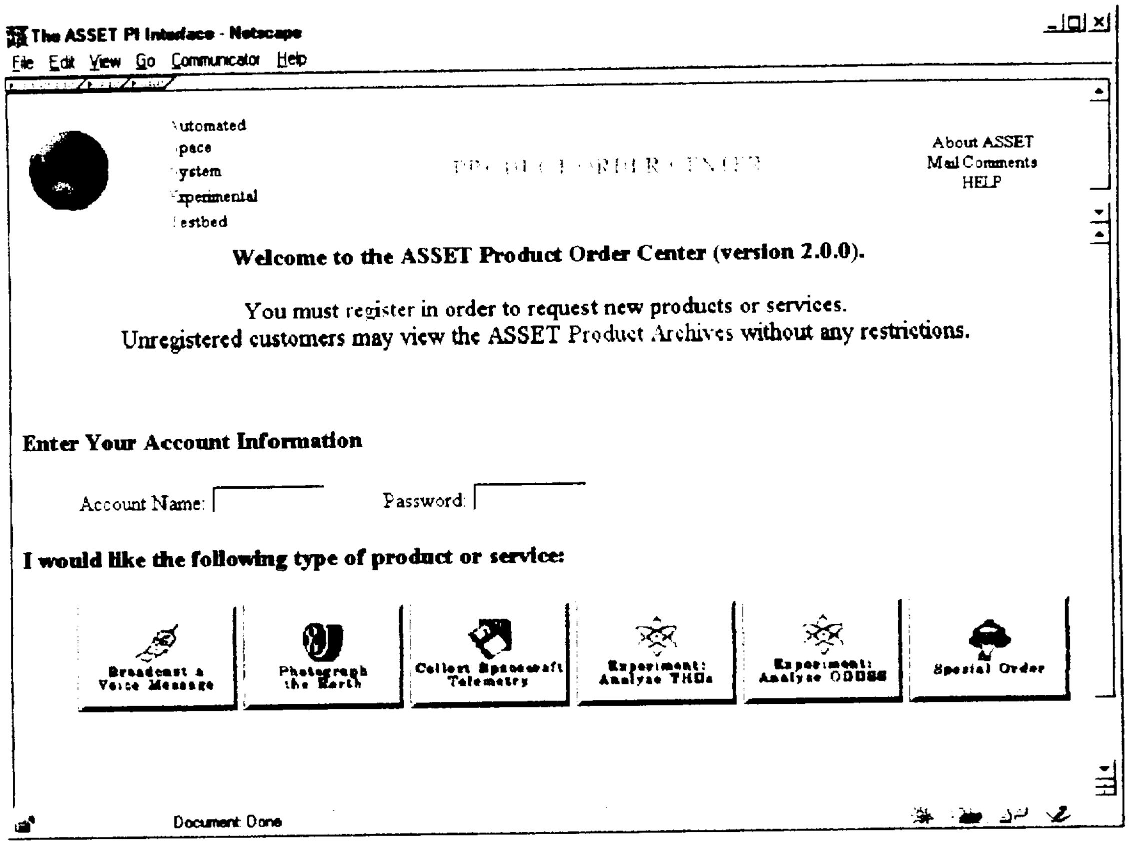
Upon loading of the client interface Web page, a product request object is instantiated by the client software. This object contains properties equivalent to the various high level product attributes and low level operational variables relevant to a general product as prescribed in the Product Specification Model. From the perspective of this object, the interview process will dictate the format of and constraints on these properties.

The first step in the client interview process, shown in Figure 5, gathers client's account information and the type of desired product. Account information is used for client authentication. Client name and the chosen product type are stored as properties in the request object; they are also used to tailor the interview process subject to the client's specification authority and chosen product type. As is seen in the figure, the first three product choices correspond to the broad product classes offered by the ASSET system. The fourth and fifth choices are special high use instances of the data collection product applied to sensor characterization of various experiments on the SAPPHIRE spacecraft. Finally, the last choice, not yet implemented, will allow direct entry of command and telemetry parameters which is essentially the lowest level of operational specification possible.

The second step of the interview gathers initial information regarding the product generation time. This information can be obtained in a variety of high or low level ways. The client chooses the desired specification method by selecting among an array of tabbed entry forms within the page. If the product target has been pre-defined within the system, then the simplest high level strategy is to choose the target from a selection list. Because the target location is a stored parameter, it is used to partially define the set of valid product generation times; in addition, choosing a pre-defined target saves time later in the interview since other target-specific attributes are already known. If the target is not pre-defined but a high level product specification is still desired, then the location of the target can be specified directly. The location for Earth surface targets may be designated by entering the latitude and longitude of the target or by clicking the appropriate region on a map. Additional location specification techniques for moving terrestrial targets, such as traveling people or vehicles, and for non-terrestrial targets, such as spacecraft, is soon to be implemented; objects of this type can be selected if they are pre-defined targets. If the client prefers to directly control the product generation time, the third entry option allows this by entering exact time constraints. Current constraint techniques include specifying the exact generation time, mandating before or prior to thresholds, and dictating a precise interest period or series of periods. A fourth technique is currently being added which will permit clients to require that product generation occur during a specific orbital event. The impetus for this is the SAPPHIRE IR experiment in which periods of interest are prescribed by observing the Earth during eclipse.

For this tour of the interface, a voice broadcast is assumed to have been chosen. Figure 6 shows step two in the interview process. The 'Pre-defined Object', 'Specific Location', and 'Period of Time' tabs can be seen across the top of the currently selected form. For this example, the client wishes to broadcast a voice message to a target audience in northern California which has not been pre-defined; accordingly, the 'Specific Location' tab has been selected. With the world map visible, the client selects the desired location and continues with the interview.

For this particular voice broadcast product, the third step of the interview is used to collect a combination of message content and target attributes. This is shown in Figure 7. Determining which of these attributes to gather occurs dynamically in response to previously entered product information. For instance, if the 'pre-defined target' specification method had been used in step two of the interview, then many of the target characteristics would already be known and would not need to be collected in the third step.



Welcome to the ASSET Product Order Center (version 2.0.0).

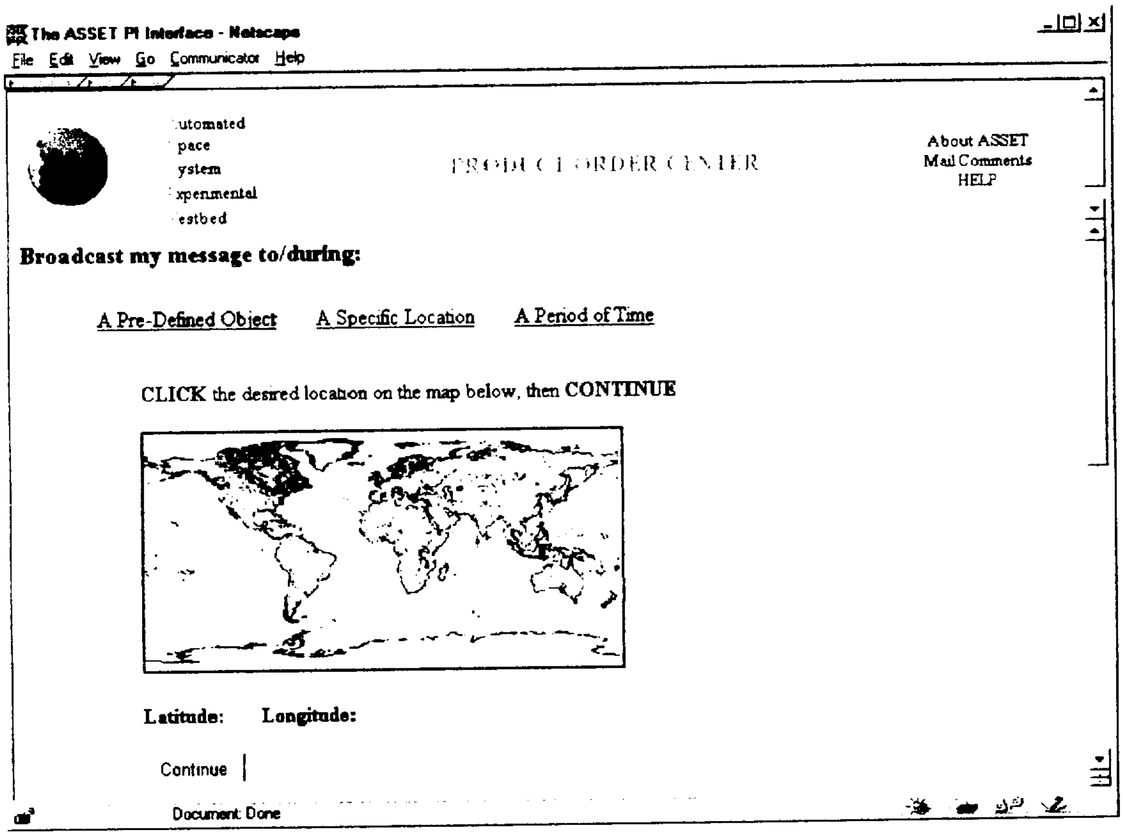
You must register in order to request new products or services.
Unregistered customers may view the ASSET Product Archives without any restrictions.

Enter Your Account Information

Account Name: Password:

I would like the following type of product or service:

Figure 5 - Specifying Client & Product Type



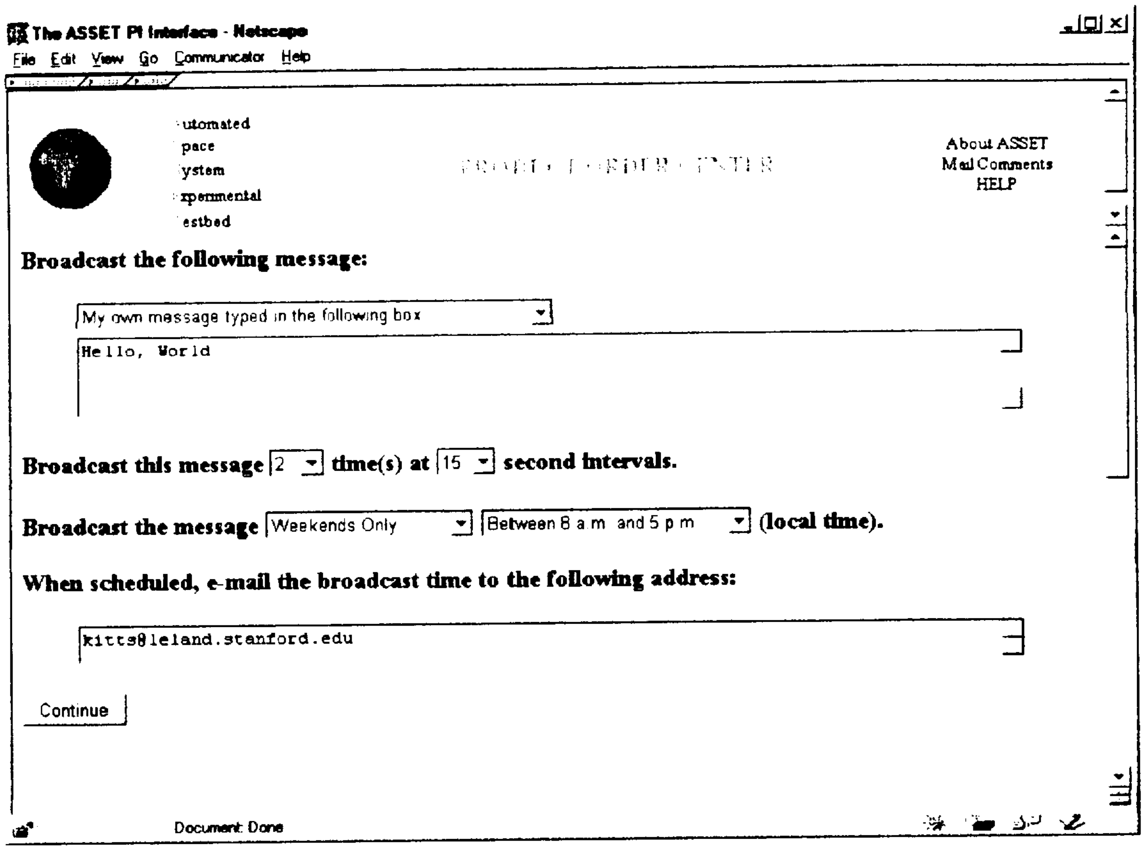
Broadcast my message to/during:

A Pre-Defined Object A Specific Location A Period of Time

CLICK the desired location on the map below, then CONTINUE

Latitude: **Longitude:**

Figure 6 - Specifying the Product's Target



Broadcast the following message:

My own message typed in the following box

Hello, World

Broadcast this message 2 **time(s) at** 15 **second intervals.**

Broadcast the message Weekends Only Between 8 a.m. and 5 p.m. **(local time).**

When scheduled, e-mail the broadcast time to the following address:

kitts@leland.stanford.edu

Figure 7 - Specifying Additional Product Attributes

Additional pages in the interview process permit additional control over many of the assumed product parameters, such as link margin, as well as other required request information, such as a cost bid for the requested product.

Upon completion of the interview, the request object is submitted to the request database within the ASSET server. Each of these is then transformed to an operational form consisting of an intersection of its constraints. A blackboard software control system is being implemented to manage additional product processing such as identifying opportunities for satisfying multiple clients with a single product, committing system resources to selected products, and integrating product command and telemetry plans into robust groundstation contact plans. Once generated, photograph and telemetry products are stored within the ASSET product database, and clients are notified of their availability for download.

6. FUTURE WORK

Near term work on this project will consist of maturing the product specification model for generality and applicability. One aspect of this will be to formalize the manner in which product-to-implementation transforms are defined so that reuse and extensibility is supported.

Development of the interface itself will concentrate on applying the principles noted in this work across the ASSET domain. In addition, more real-time processing will be migrated to the client workstation so that feedback concerning product existence and remaining degrees of freedom can be integrated within the interface. Additional upgrades include enhanced support for subject definition, client-side data validation, variable client-based specification authority, on-line HELP, real-time product costing, and a method for incrementally modifying a request prior to submission.

A more rigorous validation of the interface is also planned in the near future. Experiments and client evaluations will be conducted in order to gain both quantitative and qualitative metrics concerning the benefits of variable level specification. Particular measures will focus on interface ease of use and freedom in specifying products, improvements in the time and cost involved in processing requests, and the ability to increase the value and/or throughput of products throughout the system. The controlled, prototype nature of the ASSET system is expected to provide a unique opportunity to comparatively evaluate these system level competitive metrics.

In the long run, an even higher level of direction will be integrated into the ASSET client interface. This will support entire operational campaigns specified through simple statements concerning an overall goal. For example, specifying general interest in a hurricane could trigger a series of photographs and sensor observations from a variety of spacecraft within the system. As always, the overwhelming theme will be to field an interface allowing convenient high level specification without sacrificing the capability to exert low level control.

In addition to evolving the client interface, research will continue in order to develop a variety of other ASSET components in order to achieve more advanced and cost-efficient capabilities. This work includes investigations in planning, scheduling, anomaly management, engineering interfaces, robust execution, software architectures, and systems integration [13]. To ensure the applicability of this work, SSDL collaborates with industry and government organizations; current projects include work with NASA's New Millennium Program and Ames Research Center. Developmental progress is specifically being targeted to permit the expansion of the space system architecture so that it can accommodate more users, missions, experiments, spacecraft, and ground stations. This will certainly require the continuation of many successful partnerships with other universities. Apart from providing a more complex research testbed, this expansion will help to build a significant population of system elements in the sense that the resulting operational methodologies and models will be truly general.

7. CONCLUSIONS

The incorporation of advanced user interfaces is a strategy for increasing the competitiveness of space systems. A particularly useful quality of such interfaces is to permit varying degrees of abstraction in the process of specifying operations to be performed. High-level specification allow clients to conveniently describe the products they desire while conserving operational flexibility. Low-level specification allows clients to precisely control additional processing parameters as required by the client; due to familiarity, sophisticated clients may prefer to use this level of specification even when it affords them no increased level of control. The combination of these capabilities creates a flexible specification process that enables abstract direction without surrendering precise control authority. Preliminary results in using the ASSET client interface attest to these benefits.

In addition to improving the quality and control of services, this work supports improvements in other competitive dimensions. Use of the Product Specification Model lays the groundwork for model-based automation that can ultimately be implemented in order to reduce operational cost and to speed cycle time. Also, throughput can be increased by exploiting a complete set of operational options and by identifying opportunities for mutually satisfying multiple clients with single products. Together, these improvements contribute to making the benefits of space systems directly accessible to investigators, customers, and the public.

Overall, the ASSET system is proving to be a valuable prototype architecture for developing and validating innovations that will contribute to increasing the performance and competitiveness of future space systems. The benefit is clear: as a comprehensive, low inertia, flexible, real world validation testbed, the ASSET system will provide an unparalleled opportunity for experimentation with high risk operational technologies. Furthermore, the academic validation process will assist in supplanting anecdotal analysis commonly performed within the space community with standard evaluation practices aimed at assessing overall system competitiveness.

8. ACKNOWLEDGMENTS

The author wishes to thank current and former research students who have contributed to the development of the ASSET operations network and its client interface. Special thanks is given to current team member Mike Swartwout for his detailed review and feedback regarding this particular work. Also, the insightful comments of this paper's technical reviewers have assisted in making this a stronger contribution. In addition, appreciation is extended to Professor Robert Twiggs, the SSDL Director, for his support and guidance in the administration of this research program. The NASA Jet Propulsion Laboratory and Ames Research Center are acknowledged for their support and assistance with this research project. Finally, appreciation is extended to all of the institutions and organizations that have participated in, contributed to, and provided feedback concerning various aspects of the ASSET system. This work has been performed in partial satisfaction of graduate studies at Stanford University.

REFERENCES

[1] Christopher A. Kitts, "A Global Spacecraft Control Network for Spacecraft Autonomy Research." *Proceedings of SpaceOps '96: The Fourth International Symposium on Space Mission Operations and Ground Data Systems*, September 16-20, 1996.

[2] Christopher A. Kitts and Clemens Tillier, "A World Wide Web Interface for Automated Spacecraft Operation", *ITC/USA '96: Proceedings of 32nd Annual International Telemetering Conference*, October 28-31, 1996.

[3] Christopher A. Kitts, "Specifying Spacecraft Operations at the Product/Service Level", *Proceedings of the 47th International Astronautical Federation Congress*, October 6-10, 1997.

[4] Robert J. Twiggs and Christopher A. Kitts, "SAPPHIRE, A University Student Built Satellite for Space Experimentation", *The AMSAT Journal*, November/December 1995.

[5] Charles Bonsall (ed.), *WeberSat Users Handbook*, Center for Aerospace Technology, Weber State University, January 1991.

[6] Brian Engberg, Jeff Ota, and Jason Suchman, "The OPAL Satellite Project: Continuing the Next Generation Small Satellite Development", *Proceedings of the 9th Annual AIAA/USU Conference on Small Satellites*, September 19-22, 1995.

[7] Sean Boyle, et. al., "The Barnacle Microsatellite", in preparation.

[8] Christopher A. Kitts, *Theory and Experiments in Model-Based Space System Operations*, Draft Research Report, Space Systems Development Laboratory, Stanford University, 1997.

[9] G. M. Nijssen and T. A. Halpin, *Conceptual Schema and Relational Database Design*, New York: Prentice Hall, 1989.

[10] P. Chen, "The Entity-Relationship Model - Toward a Unified View of Data", *ACM Transactions on Database Systems 1:1, 9-36*, March 1976.

[11] Wiley J. Larson and James R. Wertz (eds.), *Space Mission Analysis and Design*, Dordrecht: Kluwer Academic Publishers, 1992.

[12] SAPPHIRE Design Team, *SAPPHIRE Engineering Documentation*, Space Systems Development Laboratory, Stanford University, December 1997.

[13] Christopher A. Kitts and Michael A. Swartwout, "Experimental Initiatives in Space Systems Operations", *Proceedings of the 1997 Annual Satellite Command, Control, and Network Management Conference*, September 3-5, 1997.

Christopher Kitts *is a doctoral candidate in and the Graduate Student Director of Stanford University's Space Systems Development Laboratory. His area of specialty is in spacecraft command and control systems. He is also an engineer with Caelum Research Corporation at NASA's Ames Research Center where he develops spacecraft design and autonomy strategies for NASA's New Millennium Program. Mr. Kitts has served in the Air Force as a mission controller of and the Chief of Academics for the Defense Satellite Communications System III spacecraft constellation. He has held a research position at the Air Force Phillips Laboratory and has taught numerous graduate courses in space system design. Mr. Kitts received a BSE from Princeton University, an MPA from the University of Colorado, and an MS from Stanford University.*

SPECIFYING SPACECRAFT OPERATIONS AT THE PRODUCT/SERVICE LEVEL

Christopher A. Kitts *
Space Systems Development Laboratory
Stanford University, Stanford, CA 94305-4035

ABSTRACT

Stanford University's Space Systems Development Laboratory (SSDL) is developing a "product level" user interface for spacecraft operations. This interface permits customers to conveniently specify the space-based products and/or services they desire in a format and language relevant to the product under consideration. Initial experimentation suggests that this style of specification will contribute to the effectiveness of space systems by 1) making these systems easier to use, 2) automating the mapping of products to procedures, 3) eliminating costly overconstraints commonly imposed through lower level specification, and 4) identifying opportunities to generate a single product that satisfies multiple customers. This interface is being incorporated into SSDL's experimental mission operations system which is being developed to conduct command and telemetry operations with several university and amateur microspacecraft. Example service offerings include Earth photography, voice broadcasting, and spacecraft data recording. At their convenience, customers specify the products they desire through the World Wide Web. This paper presents the conceptual framework for the product level interface; the incorporation of the interface within Stanford's real-world space operations system is also described.

INTRODUCTION

Increasing the economic competitiveness of space systems requires innovations to lower the cost, reduce the cycle time, and increase the quality and features of the resulting products and services. Developing and incorporating advanced user interfaces for the external customers of these systems is one particular technique for enhancing these competitive measures.

In many contemporary space systems, requesting a product or service is an inefficient process [1]. First, requests are often directed to a human mission planner in a slow, often iterative proposal process; this process often requires synchronization in time and/or space. Second, clients are often asked to specify their product at a level inappropriate to their knowledge or desire. For instance, clients only interested in obtaining the end product often must become involved with potentially complex and uninteresting details concerning the equipment and functionality of the space system. Third, the process used to specify a desired product typically relies on an experiential system for identifying a limited set of implementations that satisfy the requirements of the client; this can severely limit the flexibility of the mission planning process by overconstraining the specification of a product.

* Doctoral Candidate, Mechanical Engineering, Stanford University.

Stanford's SSDL is developing a space system client interface that addresses these problems. Speed and synchronization are addressed by automating the complete product request process and by implementing it through the World Wide Web (WWW). The result is an automated interview process through which a sufficient characterization of the desired product is captured. The process is context sensitive in order to simplify interaction. This innovation effectively allows asynchronous product requests at a time and place convenient for the client. The graphical nature and general familiarity with the WWW contributes significantly to the ease of use of this interface.

Simplicity is supported by directly collecting information relevant to the specific product or service desired. For example, a photograph may be ordered merely by specifying the subject of the photo, the deadline for delivery, and any relevant photographic parameters such as resolution, light level, etc. No knowledge concerning the capabilities or orbits of specific spacecraft, the location of communications stations, the functional status of the system, the system's command and telemetry procedures, or the nature of other product requests are required in order for the client to describe the product of interest. All operational concerns of this nature are transparent to the customer.

The elimination of overconstraints is addressed by relying on a fundamental model of the system for transforming product attributes into the set of all operational implementations that will result in the desired product. Through this technique, operational flexibility is conserved through the interface such that the full set of possible implementations is available to the planner/scheduler for consideration. Culling of this set occurs later in order to simplify or optimize the planning process; many contemporary space systems eliminate possible implementations with no regard to optimality so as to conform to organizational standards or to simplify the request process.

This interface is being incorporated into SSDL's real-world space operations system. Initial experimentation suggests that this style of specification will contribute to competitiveness by 1) making space systems easier to use, 2) automating the mapping of products to procedures, 3) eliminating costly overconstraints commonly imposed through lower level product specification, and 4) identifying opportunities to generate a single product that satisfies multiple customers.

In describing the design and implementation of the product level interface, an overview of SSDL's space operations system is first presented. Next, the conceptual technique for representing products and their implementation sets is then presented. The appearance and flow of the current product level interface is then described. Finally, future work in developing the interface is projected and general conclusions from this work are presented.

THE ASSET SYSTEM [2]

One of SSDL's primary research activities is the development and validation of new operational innovations that contribute to the system level competitiveness of space systems. In order to conduct this work in a relevant experimental environment, the Automated Space System Experimental Testbed (ASSET) system is being developed. The ASSET system is a simple yet comprehensive real-world space operations network. This system will be used to operate a variety of academic and amateur microsatellites; in doing so, it will also serve as a low inertia, flexible, real-world validation testbed for new operational methods and technologies.

Figure 1 shows a high level view of the ASSET mission architecture [3]. The basic components include the user interface, a control center, ground stations, communications links, and the target spacecraft. During the current developmental phase, a highly centralized operations strategy is being pursued with nearly all mission management decision-making executed in the control center. These tasks include experimental specification, resource allocation throughout the ground and space segment, anomaly management, contact planning, data formatting and distribution, and executive control.

Spacecraft. Three university microsatellites are currently being integrated into the ASSET system. SAPPHIRE, SSDL's first satellite, will characterize the space-based operation of experimental infrared sensors, photograph the Earth, and broadcast voice messages [4]. SAPPHIRE is currently undergoing final testing; secondary launch options are being explored. Operations with Weber State University's WeberSat spacecraft, launched in 1990, will include Earth photography and telemetry analysis [5]. WeberSat's on-board software is currently being modified to accommodate these services. SSDL's second satellite, OPAL, will test a variety of inexpensive COTS sensors and will validate a launch mechanism for deploying hockey-puck sized science craft [6]. While these spacecraft have simple missions, it is worth noting that their mission products are reasonable operational analogs to the remote imaging, direct broadcasting, and sensor recording products offered by industrial, civil, and military space systems.

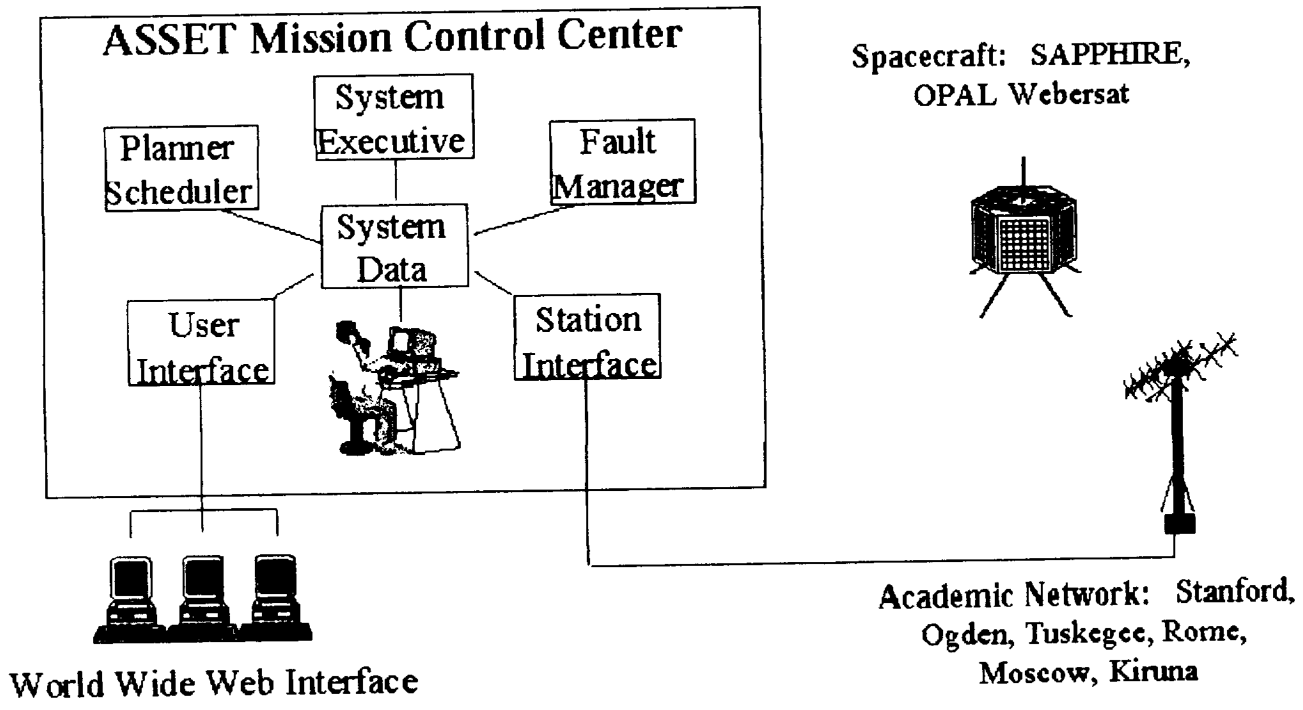

ASSET Mission Control Center
System Executive
Planner Scheduler
Fault Manager
System Data
User Interface
Station Interface
Spacecraft: SAPPHIRE, OPAL Webersat
Academic Network: Stanford, Ogden, Tuskegee, Rome, Moscow, Kiruna
World Wide Web Interface

Figure 1 *- The ASSET Space System Architecture*

Groundstations. The ASSET groundstations employ HAM radio frequencies and equipment commonly used for amateur satellite communications. Typical stations have steerable antennae and use packet radio data formats. To date, facilities at Stanford and at Weber State University have been used for experimentation. Nearly a dozen other stations throughout the world (Sweden, Italy, Russia, Japan, and across the U.S.) have been identified for future integration. The resulting network will pose planning, scheduling, and execution challenges similar to those currently experienced within the Air Force Satellite Control Network (AFSCN), the NASA Deep Space Network (DSN), and a variety of other large scale space operations systems.

Mission Control Center. The ASSET mission control center resides at Stanford University and consists of several workstations and operators/developers. In the current centralized architecture, the agents in this control center are responsible for mission planning, resource scheduling, executive control, health management, and interfacing with external users and internal engineers. SSDL's research is aimed at understanding these tasks well enough to support automated end-to-end mission products processing and systems health management.

System Processing. In the developing mission architecture, clients submit requests for products through a World Wide Web interface; these requests are stored in a central database. Various software modules filter these and system-originated health monitoring requests in order to select products for processing, schedule the spacecraft and groundstation resources required, and plan the low level contact plans necessary. Contact plans are executed via the groundstations using the spacecraft-specific command and telemetry formats. Mission products are returned to the control center for delivery to customers and for storage in a searchable archive. Telemetry is analyzed in order to detect anomalies. Anomalous conditions trigger operator notification, rescheduling of resources to support contingency activities, and a variety of diagnosis and reconfiguration agents in order to assist spacecraft engineers.

HIGH LEVEL AND INDEPENDENT SPECIFICATION

Within a space system, mission products planning can be viewed as a constrained design problem. Determining what to accomplish is a form of specifying requirements for any subsequent activities. Considering available options equates to a morphologic analysis. Selecting a specific course of action includes trading off options with respect to performance drivers. Relative to design, this process is constrained due to the short-term nature of the problem and the need to use readily available resources.

A primary benefit of specifying operations at the product level is that the resulting specification is often a far broader set of possible actions than that obtained by conventional methods. Conventional methods typically create narrower sets due to three particular phenomena [7].

First, in what is often an attempt to be helpful, clients (and occasionally system representatives) may offer false specifications; this occurs when their request is really a combination of a fundamental desire for a product combined with particular suggestions on how to produce that product. For example, a client may ask for a photograph taken by a specific spacecraft when in fact any spacecraft may be suitable for obtaining the desired photo. ASSET's client interface focuses on the attributes of the end-product itself as a strategy for eliminating this effect.

Second, given a true specification for a product, possible implementations are often overlooked due to reliance on experiential reasoning approaches. Such approaches are based upon heuristics, standard operating procedures, past experiences, and procedural knowledge bases that represent the fundamental design, behavior, and capability of the system in a very weak manner. Successful application of such systems relies heavily on the training and experience of human operators, the accuracy and timeliness of the knowledge base, and the knowledge base's span of environmental conditions and operational modes. As a result of this style of processing, possible operational implementations capable of satisfying a client can be easily overlooked. The ASSET client interface overcomes this barrier by relying on systematic, model-based methods to transform a high level specification into a complete and consistent set of operational alternatives.

Third, specifications are often inappropriately affected by the state of the space system at the time of the request. For instance, possible implementations for a new request are often ignored if they require resources that are currently unavailable due to higher priority products or system downtime. But these factors are dynamic, and resource availability is sensitive to both. As a result, small changes in the system's state can radically alter the available implementations for a particular product. The ASSET client interface tackles this challenge by seeking to capture the full set of product implementations without regard to the dynamic elements of the system's state. Instead, the ASSET system is being developed to integrate these concerns at planning-time and/or execution-time. In this manner, the system retains complete knowledge of the fundamental specification for requested services.

TRANSFORMING A PRODUCT REQUEST

In abstracting the process of specifying operations, Entity-Relationship modeling is used to formalize three specific views [8].

The Product View defines attributes and associated value ranges for relevant characteristics of the desired deliverable. This schema is ideally a system-independent view appropriate to a layman. Common product attributes include product type (i.e. a photograph, a message broadcast, etc.), an object of interest (i.e. a person, a region of the Earth, a spacecraft, etc.), type-specific parameters (i.e. resolution and light level for a photograph, content and signal quality for a broadcast, etc.), and other relevant variables.

The Processing View is a set of attributes and associated value ranges for relevant system entities and relationships. This schema is a general method for explicitly characterizing the execution-time configuration of a space system. Common system entities include product generation components (i.e. a camera payload, a transmission system, etc.), a processing subject (i.e. objects of interest for a photograph), celestial bodies (i.e. Earth, Sun, etc.), and other relevant system objects. Common relationships include distance, line-of-sight view, etc.

The Operational View is a set of attributes and associated value ranges for relevant system controls and observations. This schema encompasses all low level aspects of operating the overall space system. Common operational attributes include command parameters and sequences, telemetry verification values, equipment configurations, execution times, etc.
Analytically, the Product, Processing, and Operational views each permit representation of the same system outputs. These views

merely cast the output against different sets of fundamental dimensions in order to provide meaning from varying perspectives.

To transform a specification from one view to another, fundamental models representing the system's behavior, structure, and capability are employed. Through a combination of computation, logical processing, and search, complete and consistent transformations can be derived in a rational, systematic, and easily maintainable manner.

As an example, consider a simple version of the space-based audio broadcasting service offered through the ASSET system. A client wishes to broadcast the message "Hello, World" to friends in the northern California region within the next week. The message is to be broadcast 4 times at 1 minute intervals. To permit real-time listening, the message should be broadcast during normal working hours. Finally, the client specifies that once the exact broadcast time is determined, e-mail should be sent to potential recipients so that they will know when to listen. In this case, the client's friends have a pre-defined profile indicating their location, receiver performance standards, and other relevant data. Together, these requirements (and others they may imply) constitute a Product View as shown in Figure 2.** Conveying this request via the interface is presented later in this paper.
The resulting Processing View consists of two primary entities: a processor with audio broadcasting capability (a search yields only the SAPPHIRE audio broadcast payload) and a processing subject with an identity defined as the client's friends. As seen in Figure 3, many Processing View attributes are directly determined from specific Product View attributes or system constants/assumptions. Some require a search over the system's component space (i.e. determining the processor requires a search for all components with audio broadcasting as a defined behavior). Some are derived from model parameters or other Processing View attributes (i.e. orbital calculations).

Product Attributes	**Specified Value**
Type:	audio broadcast
Subject:	friends in northern California
Message:	"Hello, World"
# Transmissions:	4
Transmission Intervals:	1 minute
Time Constraints:	9 a.m. - 5 p.m. local time
Deadline:	one week in future
Notification:	friends@asset.stanford.edu

Figure 2 *- An Example Product View*

Processor Attributes	**Derived Values/Constraints**
Component Options:	(Components w/ a behavior = audio broadcast) = SAPPHIRE audio transmitter
Content:	From Product View = "Hello, World"
# Jobs:	From Product View = 4
Intervals btw Jobs:	From Product View = 1 minute
Deadline:	From Product View = one week in future
Position:	(Positions for Component Options Spacecraft) f(SAPPHIRE KEPS, time)
Pointing Loss:	f[attitude = f(position)]
Constants/Assumptions:	From System Model: Line Loss, Frequency, Modulation, Beamwidth, Power, etc.

Subject Attributes	**Derived Values/Constraints**
Identity:	From Product View = friends in Nor. California
Location:	From Stored Subject Profile = Nor. California
Availability:	From Product View = 9 a.m. - 5 p.m. local time
Address:	From Product View = friends@asset.stanford.edu
Required SNR:	(From Stored Subject Profile = perf. metric) ≤ (Actual SNR = f(Distance, Ptg Loss))
Constants/Assumptions:	From Stored Subject Profile: Omni, Line Loss, System Temperature, Obscura, etc.

Relationship Attributes	**Derived Values/Constraints**
Distance:	f(Processor Pos., Sub. Loc.)
Line of Sight:	f(Processor Pos., Sub. Loc., Obscura) = clear
Constants/Assumptions:	Path Loss, etc.

Figure 3 *- An Example Processing View*

** This example is similar to a high level specification for digital broadcasting service. For example, a program director might specify that a particular movie be broadcast to the Western United States sometime next week during prime time. Once the scheduling system merges this request with all other programming requests, a programming schedule is published and made available to viewers via publications, a dedicated broadcast channel, and/or the Internet.

Together, the resulting value ranges and functions constrain the operation of the system. For instance, the subject's received signal-to-noise ratio (SNR) must be above a defined minimum threshold; but all link budget parameters are fixed with the exception of distance (which varies with time) and processor pointing loss (which ultimately is also a function of time). In addition, the execution time is constrained by a deadline and a subject availability criteria. The general solution yields a set of processing time windows in which acceptable performance is possible and desired; generating the requested product outside these windows is of no value.

For Each Component Option and Host Spacecraft = (audio broadcast system on SAPPHIRE):

<u>Upload Attributes</u>	**<u>Derived Values/Constraints</u>**
Upload Time:	Intersection of (Spacecraft/Groundstation contact times) and (prior to execution time)
Groundstation:	All compatable GS = SSDL Groundstation
Groundstation Config.:	SSDL SAPPHIRE Configuration (Freq., Modem, Amps, KEPS, etc.)
Cmd/Ver Sequence:	[Schedule broadcast and verify by checking queue]
	cmd 1: **os scheduler add absolute**
	<*Exec_Time*> voice speak 4 60 "Hello, World"
	ver 1a: nominal prompt
	cmd 2: **os sched list**
	ver 2a: returned list contains:
	<*Exec_Time*> voice speak 4 60 "Hello, World"
<u>Execution Attributes</u>	**<u>Derived Values/Constraints</u>**
Execution Time:	Intersection of (time prior to deadline), (time s.t. Actual SNR > Required SNR), and (subject availability time windows)
Cmd/Ver Sequence:	[SAPPHIRE audio broadcast execution procedure]
	cmd 1: **voice speak 4 60 "Hello, World"**
	ver 1a: audible transmission "Hello, World"

Figure 4 *- An Example Operational View*

Finally, the Operational View outlines the derived procedural options that will result in the desired product. SAPPHIRE (which is the only spacecraft option for audio broadcasting) uses a general three phase operational processing model: the "upload" phase consists of a groundstation to spacecraft contact in which commands for future product generation are scheduled; the "execution" phase consists of spacecraft processing in which the commands are executed from the command queue in order to generate the product; the "download" phase consists of a groundstation to spacecraft contact in which any product files are obtained for subsequent distribution. For a SAPPHIRE audio broadcast, the upload phase is used to schedule the broadcast command for its determined execution time, the execution phase consists of the audio broadcast itself, and the download phase is unnecessary. As seen in Figure 4, Product View and Processing View attributes are ultimately converted into specific commands, telemetry checks, time constraints, configuration parameters, etc.

RAMIFICATIONS OF THE PRODUCT-OPERATIONS TRANSFORMATION

Given the existence of a transformation between views, it is worth considering the ramifications that follow from the nature of the transformation.

As has been alluded to previously, these product-to-operational implementation transformations are generally at least one-to-many. Indeed, it is this very quality that allows a product level specification to be a broader set than is commonly given by lower level specification methods. The result is an increase in system flexibility given that more options are available to the system. In general, this promotes an overall increase in the ability to generate products.

A cost of having larger implementation sets from which to choose is the increase of the ensuing search space and the escalation of the planning/scheduling problem's computational complexity. This, however, is believed to be inconsequential since the specification process itself in no way limits the planner/scheduler from pruning its search space in order to define a tractable problem. With the proposed design, this pruning becomes a rational element of the overall system rather than existing as an artifact of limited comprehension and poorly designed functional and/or organizational interfaces.

A more subtle quality of the product-to-operational implementation transformation is a potential many-to-one characteristic. That is, a single operational implementation may generate an output that qualifies as an acceptable product for multiple clients. For example, the same photograph of a region on the earth might have value for a resident of that region, a scientist gathering photographs of the entire earth, and a spacecraft engineer wishing to test and calibrate the camera.

By providing the high level specification of products, the ASSET system is being designed to promote, identify, and take advantage of such many-to-one opportunities in order to increase throughput at little or no additional operational cost. This is being accomplished by implementing a planning/scheduling preprocessor that identifies intersections between the implementation sets of different products. For instance, consider two product requests, one high priority and one low priority. Let the implementation set for the high priority request be a proper subset of the other's implementation set. In a situation such as this, satisfying the high priority request will always satisfy the lower priority request as well. The preprocessor identifies situations such as these by removing the lower priority request from the planner/scheduler's attention under the condition that the higher priority request can be satisfied.

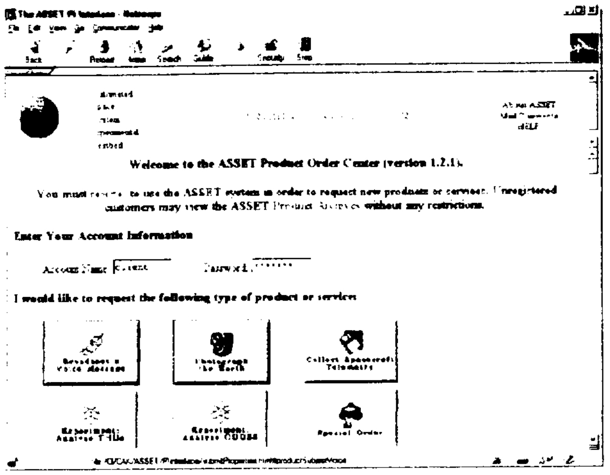
THE PRODUCT LEVEL INTERFACE

The methods presented in this paper for specifying and transforming product level requests are being integrated into the ASSET space system. The client interface collects relevant product attributes and ultimately submits these to the ASSET database. These specifications are then transformed to their operational form. The planning/scheduling preprocessor identifies opportunities for generating single products in order to satisfy multiple requests. The planner/scheduler then commits system resources to selected products. The ASSET execution system is being designed to process the low level system processing in a robust manner.

The client interface itself exists as a WWW page written in HyperText Markup Language (HTML) and JavaScript. Upon request, the ASSET Web server relays the proper files to the client's Web browser. For the previously discussed example, the first step in the client interview process, shown in Figure 5, is to obtain the client's account information and the type of desired product. Once entered, the account information is relayed back to the ASSET server for authentication.

In the second step, the product's subject is specified as is shown in Figure 6. A selection list exists for designating pre-defined subjects that have their attributes stored within the ASSET database. Alternatively, the client may specify a region of the Earth as the subject's location by simply clicking on the appropriate map location. In general, the interface will allow simple specification of Earth-fixed subjects (i.e. facilities, territories, etc.), traveling subjects (i.e. people, vehicles, storm systems, etc.), and space objects (i.e. celestial bodies, spacecraft, etc.).

In the third step, additional product attributes are collected as displayed in Figure 7. The decision concerning which attributes to gather information on occurs dynamically in response to previously entered product information. Additional attributes are collected on subsequent pages of the interface.


Back
Reload
Home
Search
Guide
Security
Stop
About ASSET
Mail Comments
HELP

Welcome to the ASSET Product Order Center (version 1.2.1).

You must register to use the ASSET system in order to request new products or services. Unregistered customers may view the ASSET Product Archives without any restrictions.

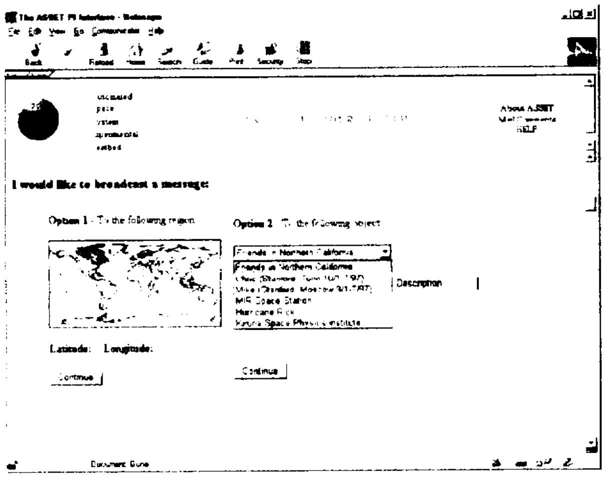
Enter Your Account Information

Account Name: client Password: [illegible]

I would like to request the following type of product or service:

Broadcast a Voice Message
Photograph the Earth
Collect Spacecraft Telemetry
Experiment: Analyze TBlle
Experiment: Analyze GOOBS
Special Order

Figure 5 - *Specifying Client & Product Type*



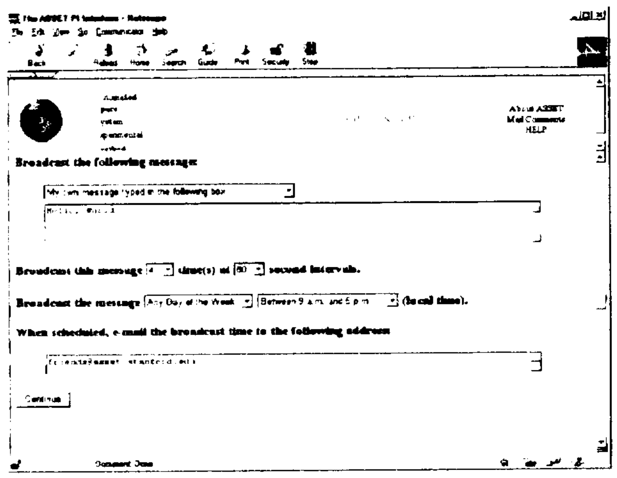
I would like to broadcast a message:

Option 1 - To the following region: Option 2 - To the following subject:

Latitude: Longitude:

Friends in Northern California
Friends in Northern California
Chris (Stanford, Turin 10/7-1/97)
Mike (Stanford, Moscow 9/1-7/97)
MIR Space Station
Hurricane Rick
Kyoto Space Physics Institute

Figure 6 - *Specifying the Product's Subject*



Broadcast the following message:

My own message typed in the following box

Hello, World!

Broadcast this message 4 time(s) at 60 second intervals.

Broadcast the message Any Day of the Week Between 9 a.m. and 5 p.m. (local time).

When scheduled, e-mail the broadcast time to the following address:

friends@asset.stanford.edu

Figure 7 - *Specifying Additional Attributes*

Previous ASSET interfaces relied on early HTML versions and were implemented prior to the availability of JavaScript or other similar scripting languages. These early interfaces relied on server processing at each stage in the interview process. Introducing JavaScript improves this by speeding processing for the client, requiring less ASSET server resources, and providing a convenient means to track the state of a request (i.e. no need for temporary server-side files, user identification codes, client-side cookie files, etc.)

FUTURE WORK

A variety of upgrades are planned in the near term for the current product level interface. These include additional techniques for subject definition, client-side data validation, a HELP system, real-time product costing, and a method for incrementally modifying a request prior to submission. A major alteration of the interface software is being planned in order to take advantages of the evolving Dynamic HTML (DHTML) 4.x specification. Using new language constructs combined with JavaScript, DHTML will allow the creation of compact interface software capable of dynamically modifying the interface without server intervention. This capability will be used to conveniently provide context-sensitive forms and help, state-based request modification, and client-based access to particular specification functions.

With sufficient knowledge acquired from developing the product level interface, SSDL will soon begin developing a new interface allowing the specification of space system operations across a spectrum of abstraction. At the lowest level, clients with sufficient knowledge will be able to conveniently specify the precise command and telemetry parameters required in order to achieve their objectives. At a higher level, an evolved version of the current product level interface will allow specification of desired product attributes. At the highest level, entire operational campaigns will be specified through simple statements concerning an overall goal. For example, specifying general interest in a hurricane could trigger a series of photographs and sensor observations from a variety of spacecraft within the system. Between these three points in the specification spectrum, a series of discrete intermediate steps would permit a client to choose the most convenient and applicable technique. The overwhelming theme for an interface of style is to allow specification at a level high enough to be convenient but low enough to ensure the desired level of control.

In addition to evolving the client interface, research will continue in order to develop a variety of other ASSET components in order to achieve more advanced and cost-efficient capabilities. This work includes investigations in planning, scheduling, anomaly management, robust execution, software architecture, systems engineering, and concurrent design. Developmental progress is specifically being targeted to permit the expansion of the space system architecture so that it can accommodate more users, missions, experiments, spacecraft, and ground stations. In addition, experiments aimed at increasing the level of autonomy, migrating control authority throughout the system, and applying new reasoning approaches are particular objectives of the current research team.

A prime element of this experimental work consists of competitively validating developed innovations. By incorporating new techniques directly into the ASSET operations system, comparative analyses concerning cost, time, and quality of service can be performed in order to measure the true contribution of a technique in a controlled environment. To further ensure the applicability of this work, SSDL will continue to cooperate with industry and governmental space organizations. These collaborations include work with NASA's New Millennium Program, Deep Space Network, and Ames Research Center. Current projects include experimentally validating a health monitoring beacon system, developing strategies to summarize spacecraft data, and determining the value of advanced user interfaces. Finally, SSDL hopes to build upon previous cooperative activities with a variety of academic institutions throughout the world in order to expand the ASSET system for both cost-efficient operational services and system experimentation.

CONCLUSION

The incorporation of advanced user interfaces is a strategy for increasing the competitiveness of information-based products and services. This is especially true for space systems due to their complexity and distributed nature. Initial work with the ASSET space system's product level interface suggests that intelligent interfaces can be used to increase the quality of services by making the request process simple, convenient, and in a high level language suitable to the client. In addition, operations costs can be reduced due to automation. Finally, throughput can be increased by exploiting a complete set of operational options and by identifying opportunities for mutually satisfying multiple clients with single products. This work is directly supporting NASA's vision of "science from the laptop" in order to make the benefits of space systems directly accessible to investigators, customers, and the public.

The ASSET system is proving to be a valuable prototype architecture for developing and validating innovations that will contribute to the performance and competitiveness of future space systems. The benefit is clear: as a comprehensive, low inertia, flexible, real world validation testbed, the ASSET system will provide an unparalleled opportunity for experimentation with high risk operational technologies. Furthermore, the academic validation process will assist in supplanting anecdotal analysis commonly performed within the space community with standard evaluation practices aimed at assessing overall system competitiveness.

ACKNOWLEDGMENTS

The author wishes to thank current and former research students who have contributed to the development of the ASSET operations network and its client interface. Special thanks is given to current team member Mike Swartwout for his detailed review and feedback regarding this particular work. In addition, appreciation is extended to Professor Robert Twiggs, the SSDL Director, for his support and guidance in the administration of this research program. The NASA Jet Propulsion Laboratory and Ames Research Center are acknowledged for their support and assistance with this research project. Finally, appreciation is extended to all of the institutions and organizations that have participated in, contributed to, and provided feedback concerning various aspects of the ASSET system. This work has been performed in partial satisfaction of graduate studies at Stanford University.

REFERENCES

[1] Kitts, Christopher A., and Clemens Tillier, "A World Wide Web Interface for Automated Spacecraft Operation", ITC/USA '96: Proceedings of 32nd Annual International Telemetering Conference, October 28-31, 1996.

[2] Kitts, Christopher A., and Michael A. Swartwout, "Experimental Initiatives in Space Systems Operations", pending publication in the Proceedings of the 1997 Annual Satellite Command, Control, and Network Management Conference, September 3-5, 1997, Reston, VA.

[3] Kitts, Christopher A., "A Global Spacecraft Control Network for Spacecraft Autonomy Research." Proceedings of SpaceOps '96: The Fourth International Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, September 16-20, 1996.

[4] Twiggs, Robert J., and Christopher A. Kitts, "SAPPHIRE, A University Student Built Satellite for Space Experimentation", The AMSAT Journal, November/December 1995.

[5] Bonsall, Charles, ed., WeberSat Users Handbook, Edition 1.0, Center for Aerospace Technology, Weber State University, January 1991.

[6] Engberg, Brian, Jeff Ota, and Jason Suchman, "The OPAL Satellite Project: Continuing the Next Generation Small Satellite Development", Proceedings of the 9th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 19-22, 1995.

[7] Kitts, Christopher A., Theory and Experiments in Model-Based Space System Operations, Draft Research Report, Space Systems Development Laboratory, Stanford University, 1997.

[8] Chen P., "The Entity-Relationship Model - Toward a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, No 1, March 1976, pp 9-36)

The Design of a Highly Configurable, Reusable Operating System for Testbed Satellites

Student Author: Rajesh K. Batra[*]
Dept. of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305-4035

Faculty Advisor: Robert J. Twiggs[†]
Space Systems Development Laboratory
Dept. of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305-4035

1 Introduction

Quick testbed satellites, typically inexpensive, built in quantity, and designed to run customer specified experiments offer researchers the opportunity to test, in a relatively inexpensive manner, experiments in space. Likewise, quick testbed satellites offer companies the ability to prototype new products in a relatively short design cycle.

The design of an operating system (O/S) for a quick testbed satellite is different from ordinary operating systems found on today's satellites. In the case of a typical satellite, the mission is specific and therefore the software on board is designed with that mission in mind. For a testbed, the satellite bus has been designed ahead of time in a few standard configurations. A researcher would select a bus that most satisfies his needs. Therefore, it is important to note that the experiments are not known up-front. Rather, interested researchers identify their experiments at a later time. Since there can be multiple researchers, the experiments may come in at various times prior to launch.

Unable to identity the experiment up-front, or whether a particular researcher can construct his experiment on time before launch can produce many problems for the testbed provider. If the testbed satellite is dependent upon one experimenter who did not complete the experiment on time, the satellite is forced to either fly with one less payload or is forced to delay the launch. In either case, the dependency causes financial loss. In addition, not knowing the experiment up-front, or not having the payload hardware available can cause the software engineer, whose job is to interface the payload to the command and control software, to begin programming and debugging late in the design cycle. This rush can lead to software that is not fully tested and as history has shown can lead to the demise of entire satellites.

What we propose is the design of an *extendible, reusable* operating system that is easily configurable to both the customer's specific mission and to a specific satellite. Since this O/S is highly configurable, there is nothing to stop it from running on a personal computer and acting as a satellite simulator. A researcher, armed with this simulator, could develop and integrate his payload completely independently of other customers and the satellite manufacturer. The testbed provider is now no longer dependent on a particular researcher. Those customers, who have completed their payload before launch, can quickly integrate their payload and *tested* software into the satellite. By severing the dependency of the payload provider from the testbed provider, we have made the design cycle more efficient, and hence cost-effective.

For proof of concept, we have designed a configurable O/S that has been implemented on both a SQuiRT (Satellite Quick Research Testbed) class satellite known as SAPPHIRE [1, 2] and on a personal computer. The operating system, known as Chatterbox, has been designed independent of the customers' specific missions. A common interface and basic facilities (to be described below) are available to all customers on all SQuiRT satellites. Therefore, it is not important for a customer's payload to be on a particular satellite [1]. In fact, in future missions, we encourage multiple customers to develop their experiments in tandem. Those that complete their payload in time would make the coming launch, otherwise, customers

[*]Graduate Student, Aeronautics and Astronautics Department. Email: raj@kaos.stanford.edu

[†]Professor, Aeronautics and Astronautics Department. Director of Space Systems Development Laboratory. Email: btwiggs@leland.stanford.edu

[1]We assume for simplicity that each satellite will be launched in the customer's target orbit.

will have the opportunity to launch on the next available SQuiRT satellite.

The intent of this paper is to discuss some of the common elements needed for constructing a reusable, configurable operating system (Section 2), the user interface for such an O/S (Section 3) and finally how to architect such a system (Section 4). Along the way, the benefits and effectiveness of the system will be discussed in greater detail; and the process required to configure the O/S for a specific mission will be described.

2 Facilities provided by a testbed O/S

Every customer be it researcher or corporation that runs an experiment or tests a prototype on a satellite is concerned with the following issues:

1. health of the satellite;
2. feedback from their payload;
3. notification of any payload errors/malfunctions.

Other facilities that may be of interest to the customer:

1. Active control over payload;
2. Data storage acquired from payload and transmission of data to ground;
3. Security and privacy of payload and acquired data;
4. Time base activation of payload.

These concerns listed above remain the same from satellite to satellite. So rather than rewrite code which can potentially introduce bugs and requires additional testing, have the operating system be responsible for providing these facilities. If written correctly, the operating system can be reused on other satellites. The customer's responsibility is merely to exploit the facilities provided by the O/S. The benefits of this open architecture design is tremendous. Payloads can design both hardware and software independent of the satellite bus.

Satellite simulators could be distributed to all customers. This would allow the individual customers to test their payload and corresponding software prior to integration with the real satellite. By testing ahead of time, this could produce huge cost savings. The company would not have to send personnel on-site to a bus provider for large periods of time, nor would the company be restricted to the time constraints of a particular satellite launch. Testing and debugging would be restricted solely to the experimenter.

The facilities in the above lists, including a portable satellite simulator have been developed for the SAPPHIRE satellite and are discussed in detail below.

2.1 Health monitoring

The health of a satellite and its payloads is critical to a mission's success. Each component of the satellite has operational limits. If a limit is exceeded, various contingencies can take place. At a minimum, an error or warning is logged, leaving the responsibility of the component's welfare to the ground-station crew. At a maximum, the satellite may reset or completely shutdown the faulty component.

The engineers that design a particular payload or component know it best; they are cognizant of the operational limits and actions required to minimize damage. Therefore, allowing the customers the capability to program the parameter limits and to test the health of their payload would be best.

SAPPHIRE has implemented a simple to use health table system reminiscent of a spreadsheet. Each row of a table is composed of four entries. The first entry is the test variable; the second and third entries are the allowable upper and lower limits on the test variable; and the last entry specifies where to branch when the test variable falls above or below the upper or lower limits. For example, suppose a particular payload is powered from two sources, a primary and secondary source. At least one source must provide a minimum of 10 volts for the payload to operate normally. Sensors are connected to both power sources. A possible implementation would be to use two tables. The first table's test variable reads the voltage from the primary sensor. The lower limit is set to 10; and the branch is set to launch the second table. Table 2 has entries similar to the first table. Table two's test variable reads secondary sensor, and the branch is to an error logging and correcting function (see Section 2.3). So if the voltage falls below 10 volts on sensor 2 an error function is called. Otherwise control is returned back to table 1.

The use of tables provides a flexible, and configurable mechanism to manage the health of a satellite. A major advantage to using tables is that they can be reprogrammed in space. This may be necessary as batteries begin to deteriorate. Voltage limits may need to be relaxed. Another benefit is that table entries can be prioritized. For example, in the case of power management, non-critical payloads can be listed first in the table. These payloads would then be powered off before any others, potentially providing the satellite the required power to sustain itself.

The health monitoring system is tied to the beacon system [3]. This provides a way for the satellite or payload to call for help to anyone with a basic receiver. There is no need to log onto the satellite to check for faults. So for example, a corrective action to a payload may be to shut it down and to broadcast a "payload failure" message.

2.2 Feedback and control facilities

An O/S should provide an interface to the satellite bus's input and output ports. Since the O/S is portable, the communication layer to these ports should be abstract enough to allow for future growth in bandwidth and number of ports. On the SAPPHIRE satellite, we have provided 32 digital input/output lines primarily used for powering on and off payloads. In addition we have provided 32 analog to digital channels for measurements and four serial ports with configurable baud rates. Future satellites may provide more serial ports, higher bandwidth parallel ports, or more control lines. The function call syntax to the O/S would remain unchanged. As mentioned in the health monitoring section, we have tied the A2D channels to the O/S's health monitoring facilities. Based on limits of these channels, corrective actions maybe taken.

Once the O/S provides a conduit to the hardware, a payload provider merely needs to write interfacing code to their particular payload. In the personal computer world, this is known as a device driver. We have designed SAPPHIRE's operating system, Chatterbox, to be easily extendible. The customer can quickly attach his device driver and commands to operate his payload. He can fully test the feasibility of his design on the satellite simulator. Once he is assured of the operations, the code and payload can quickly be integrated into the flight satellite. This provides a huge cost savings to both the payload and satellite bus providers. An in-depth discussion of Chatterbox's interfacing language is explained below.

2.3 Error logging and correcting facilities

An operating system designed for testbed payloads must provide an error reporting mechanism accessible to all payloads. By having the O/S manage the errors, we guarantee consistent error reporting, and provide corrective actions at a level that may not be available to the payload provider. For example, if a payload reports a general error message stating that it needs more power, the O/S maybe able to turn off non- critical payloads to solve the problem. Having the O/S manage and make decisions over the satellite concerns, payloads are able to submit complaints and have them resolved in a safe manner.

2.4 File system

Chances are high that a researcher using a testbed satellite would like to store and retrieve measured data. Therefore, an O/S should provide a mechanism for creating files, similar to O/S's found on ground based personal computers. The files could be stored in RAM or space designed disk-drives. However the means for storage, the O/S should provide the file system abstraction. The benefits to a file structure are numerous: All data on the satellite, regardless of the payload would be handled the same way. For example, the retrieval of a file to Earth would be the same for a payload that has acquired a photograph and for a payload that has measured data from a horizon detector. A file system provides information such as creation time, data size, and owner. A file system can also provide security over data. This could allow for multiple experiments by different parties to be on the same satellite and would ensure privacy of the results for the respective customers. At a lower level, a file system is beneficial for memory management. By having this layer of indirection, we have the benefit of localizing all memory allocations, which then can be replaced with more robust memory algorithms for space environments, such as software error detection and correction (EDAC).

2.5 Simultaneous multi-user support

There are several management methods for handling multiple customers on a testbed satellite. The most restrictive would be to have the satellite provider be in charge of all operations on the satellite. This would require the payload provider to either send personnel to the satellite operation facilities or to train the satellite provider's team on the operations of the payload. This is neither the most efficient time or cost solution.

A solution that we have adopted is to design an easy to use operating system which allows each customer to construct their own commands to operate the satellite. This would allow the customer to operate their own payload and to log on directly to the satellite. Since the commands are designed by the customer and can be fully tested using a satellite simulator, the customer benefits by being able to simulate operating procedures months before the launch. Also, the payload provider now has the option of not needing to send support personnel to the satellite manufacturer, provided that the payload company has a ground station. Allowing customers the freedom to manage their own payload adds the possibility that multiple customers may want to access the satellite simultaneously. Therefore, an operating system which supports multiple users is preferred. We have designed the SAPPHIRE satellite to allow up to 26 users simultaneous access. The Chatterbox operating system allows as many users as bandwidth and memory allow.

2.6 Scheduler

Researchers are often interested in acquiring data when the satellite passes over a particular region of the Earth. Chances are high that they will not be logged in at that time. Either the researcher needs to network with more ground stations or have the operating system execute user specified commands at certain times. The Chatterbox operating system has been designed with a scheduler system. Commands, specified by the researcher, are executed with the researchers access rights and privileges at a future time. Commands can be added, listed, or removed from the scheduler.

Another solution that we are studying to simplify the interaction with testbed satellites is to have a customer use the world-wide web to request operations to be executed on the satellite [4]. Remote access would remove the necessity of sending personnel to particular site locations. In addition, by having enough ground stations networked, almost continual coverage of the satellite can be attained. This would, additionally, free the customer of operation costs for a groundstation. Chatterbox's interface has been designed with the notion of eventually using automated scripts. See next section below.

3 User interface

The interaction of a user with a satellite is by far one of the most important issues to consider. A confusing interface can lead to severe errors and at best require unnecessary training and a guaranteed obsolete operating system. No matter how intricate the design of the operating system, if the interface is clumsy, only under duress will a user be willing to learn the system. Is it no wonder that systems have moved from command line to graphical user interfaces (GUI). And even in the windowing environment, we see continual improvements. It has been the author's experience that satellite interfaces are reminiscent to interfaces similar to the Altair 8800 (merely switches), and at best, systems such as DOS, UNIX or VMS which has a hodgepodge of commands lacking a consistent thread[2]. Of course, installing a GUI interface on a satellite at this point in time would be overkill and a waste of bandwidth. Only when the number of users increase to a high enough level, that it becomes more efficient for an O/S to be obvious than to expend the resources in training would it merit going to a graphical interface. And, the first logical GUI interface would be implemented on the ground and merely translate the mouse clicks to actual satellite commands. However, even at this level it pays to have a well structured O/S command set.

An operating system interface for a testbed satellite varies from a conventional O/S in that the login time is limited to very short periods. For SAPPHIRE, a typical login window is about 15 minutes. Therefore, an efficient command set is necessary. Actions need to be correctly executed using the minimum description from the user.

Regardless, an O/S should be easy to learn and to a certain extent intuitive. This design criteria will have huge payoffs when an unexpected contingency occurs. An obvious command set may actually save the satellite or payload from a customer who had not implemented complete emergency procedure. If a command is difficult to recall or its semantics inconsistent, chances are reduced of successfully applying an action. The vocabulary should be reused consistently. Language defines and bounds our thoughts and ideas. If the semantics of an O/S interface are limited or inconsistent, the ideas and actions will be too. Ideally, basic behaviors or words should be combinable to form sophisticated behaviors or thoughts. Additional benefits of a reusing vocabulary is that there are less commands for a customer to learn, and typically makes the software implementation more compact.

3.1 Chatterbox user interface

For the SAPPHIRE satellite, a sophisticated language requiring a grammatical parser was beyond the

[2]The author is currently a system administrator (on the side) for a cluster of Sun and Silicon Graphics workstations at Stanford University, and is proficient with several (out)dated O/S's including VMS, and AmigaDOS. In no way is he attempting to malign the progress that been made in the history of operating systems He merely acknowledges that as with most things, improvements are always forthcoming.

Component	Description
voice	Addresses voice synthesizer payload
camera	Addresses the camera payload
sensor	Addresses telemetry & IR payload
os	Addresses system commands
help	Provides help on any component

Table 1: Addressable components on SAPPHIRE.

Action	Description
acquire	Activate payload, spools data to file.
list	lists all files associated with Component
get	sends file to ground
delete	removes file
view	Activate payload & spool data to screen
speak	Enunciates given string to ground

Table 2: Actions for voice, camera and sensor components.

scope of its mission timeline. However, a consistent vocabulary and syntax was defined. The first parameter(s) describe which component an action will be applied to, i.e. a noun. The next parameter is the explicit action (verb) followed by any modifiers. So command sentence would be constructed as follows:

(Component) (Action) [optional modifiers]

For SAPPHIRE, the payloads were known up front. They are a voice synthesizer for enunciating user sentences to ground, a camera for taking pictures, and infrared sensors. There are also housekeeping commands such as viewing logged errors, viewing the current time, etc. that are compartmentalized under system commands. Note, for ease of use, the help component which is clearly a part of the system has been moved up to the top level for immediate access. Therefore there are five main components for SAPPHIRE: voice, camera, sensor, os, and help. A description of each is listed in Table 1.

The number of actions for the payloads were surprising small. They are listed in Table 2. Note that if a customer is providing a new payload, his job would be to create a component name, and any new action words required to operate the payload. For the voice synthesizer, the action word speak became necessary and was added solely to the voice component vocabulary.

A few examples will now be given to clarify Chatterbox's usage: To take a picture, or to acquire data for channels 1 through 10, the commands are very similar:

camera acquire
sensor acquire 1-10

Notice for the sensor command, a modifier indicating which channels to acquire the data was specified. Additional modifiers such as sample rate and duration could have been specified, but instead the default values were accepted. To list what files the payloads have created, the following similar commands are used:

camera list

```
Entry #   Size (Bytes)   Owner    Date
1         22,000         kb8tfa   12:15 6/6/97
2         12,000         kb8tfa   12:20 6/6/97
```

sensor list

```
Entry #   Size (Bytes)   Owner    Date
1         10000          ke6qmd   13:15 6/5/97
```

To retrieve the files to ground, one merely addresses the component with the get command and specifies which entry they wish to get.

camera get 1,2
sensor get 1

In a similar vein, a basic vocabulary which is reused often has been applied to the entire satellite. Since this paper is not intended to be a users guide for operating the Chatterbox O/S, a complete description of all system commands will not be discussed. Parties interested in a detailed implementation of Chatterbox may contact the author directly.

4 Architecture for implementation of a testbed O/S: Modular design

We have given the customer the freedom to design their payload, write the required software drivers (see section 2.2), and to test the software/payload interaction independently of the satellite. As stated previously, a satellite simulator capable of running on most personal computers is provided to the customer. Once everything is operational, the payload and software can be installed on the next available satellite. This section discusses the elements required to successfully implement an O/S capable of being ported to various platforms, and a framework for allowing user customization.

At the highest level, the O/S should be broken up into fundamental units known as a modules. Similar to an atom which is composed of neutrons, protons, and electrons, a module can be decomposed into an interface, and implementation section. The implementation section is a body of code that performs the behavior (or operation) of the module; it is encapsulated (or hidden). The interface section is the front end of a module; it is a liaison between the user and the implementation section. The interface should never change. It is in the coarsest sense the language needed by the user to have the module perform an action. Changing the language affects the user directly. However, the implementation can and often times will change. Typically code is rewritten to optimize certain operations. Since the implementation is hidden behind the interface, changes will not adversely affect the user. Examples of some of SAPPHIRE's optimized implementations are given below.

There are many benefits to creating modules, we will list only a few of these benefits. Modules can be portable and most certainly reconfigurable. To realize the benefits of *portable and reconfigurable* requires a bit of background explanation. A properly designed module can be categorized as Machine Independent (MI), Device Independent (DI), or Machine and Device Independent (MI/DI). A machine is defined to be the hardware on the satellite on which the O/S is run. It is typically called the CPU (central processing unit), and varies from satellite to satellite. A device is a peripheral added to the satellite. For example, a customer payload is a device. The corresponding software required to operate (or drive) the device is a device driver. So, a Machine Independent module is code that is completely independent of the actual CPU that it is running on [3]. For example, a module that controls a customer's payload is MI. Even though a customer may use commands from another module to communicate with the RS232 port; he is accessing the interface of another module (in this case, a machine dependent module). Regardless of what other modules the customer may access, so long as he never calls machine specific code directly, his module will remain Machine Independent. Of course, since his code is designed to communicate with his payload, it is not Device Independent. An example of a MI/DI code, is an algorithm that sorts numbers (such as a quick sort). Now, with that explanation in hand, machine portable code is code that is Machine Independent (MI). Hence, device drivers are portable modules. A module can be reconfigured by merely changing the implementation section. So, suppose we have a RS232 communication module designed to run on the Motorola processor. Now, suppose we want to run this module on a simulator that uses an Intel processor. The interface code the customer uses to receive and transmit data remains unchanged; all that is required is to change the implementation section to Intel specific code. By changing just this one module, all MI code above will automatically work on the simulator. We have saved a tremendous amount of work which directly translates to cost savings. Another benefit to modules is that they can be shut down or replaced easily. If a payload is inoperable or will be replaced by another payload the corresponding software module (and the calls to that module) need be replaced. Note, that there is some effort involved; however, modules try to minimize that effort. In order to communicate with the Chatterbox O/S, the first parameter required was the component (see Section 3.1). From an implementation point of view, a component is nothing more than a module. If the Camera payload fails in space, all commands addressing the camera module can be shutdown with no affect on the other components.

Modules can have access levels. Since payloads are owned by various customers, a certain amount of privacy may be required for their data. Just as a module can be shutdown, a module can limit the access to certain parties. On the SAPPHIRE satellite, the IR sensors are a Jet Propulsion Laboratory (JPL) experiment and only authorized personnel have access to the sensor controls.

Modules can be optimized and designed in tandem. Since the implementation is encapsulated, a customer can make sweeping changes to increase speed or minimize memory usage without affecting the entire system. On the SAPPHIRE satellite, the IR sensors were originally required to acquire data at 4 Hz. In time, the specification changed to 100 Hz. This required substantial modifications to the implementation; however, once implemented, it was transparent to the ground station personnel. Customers can design various modules in tandem. As mentioned before, multiple payloads can be built and added to the framework.

5 Additional thoughts

The concept of modular design has been preached throughout academia and has been implemented with varying degrees of success in particular industries. However, a certain amount of sloppiness is inherent to a project of any substantial level. Satellite fabrication, as most would agree, is a project of large magnitude requiring the cooperation of many people, specialized in many fields, across various industries. In order to reduce some of the inherent sloppiness in a system, a certain amount of rigor must be asserted

[3] The author assumes the language the code is written in is Machine Independent (i.e. ADA, C, C++ not Assembly/machine language)

in the language and protocol. Object-oriented programming languages have been developed to enforce this modularity notion and to aid in removing some of the inherent sloppiness found in procedural languages. However, the author would like to entertain the idea that not only should software modules exhibit behavior, but in fact so should hardware. If hardware is designed with an interface, the hardware can be requested to go off on a task and notify the requester of its completion. Adding this level of autonomy has just modularized the entire design process, and has given the capability of replacing hardware devices with more optimized devices without affecting the entire system. The author concedes that establishing the protocols, and design specifications for customers to follow is a substantial task. But in the long run, a standardization will produce far more benefits and advancements to satellite technology.

Acknowledgments

I would like to thank Kenneth P. Koller for his extraordinary effort in helping me to focus the design, sharpen the vocabulary, and implement the Chatterbox O/S. The actual name for our O/S came about one Sunday morning as Ken and I drove up to campus listening to Garrison Keillor's Prairie Home Companion Theater. When we found out that Pete's Pretty Good Groceries was right across from the Chatterbox Café, we knew we were on to something.

About the author

Rajesh K. Batra: Graduate Research Assistant at the Aeronautics and Astronautics Department at Stanford University. He received his BSc in 1993 in Aerospace Engineering at the University of Cincinnati, and his MSc in 1994 in Aeronautics and Astronautics from Stanford University. Rajesh K. Batra has extensive experience in object oriented design and simulation. He has consulted in industry in the area of software design and is the primary operating system designer for the SAPPHIRE satellite.

References

[1] C. A. Kitts and R. J. Twiggs, "The satellite quick research testbed (squirt) program," in *Proceedings of the 8th Annual AIAA/USU Conference on Small Satellites*, Aug. 1994.

[2] C. A. Kitts and W. H. Kim, "The design and construction of the stanford audio phonic photographic infrared experiment (sapphire) satellite," in *Proceedings of the 8th Annual AIAA/USU Conference on Small Satellites*, Aug. 1994.

[3] M. A. Swartwout and C. A. Kitts, "A beacon monitoring system for automated fault management operations," in *Proceedings of the 10th Annual AIAA/USU Small Satellite Conference*, Sept. 1996.

[4] C. A. Kitts and C. Tillier, "A world wide web interface for automated spacecraft operation," in *Proceedings of 32nd Annual International Telemetering Conference*, Oct. 1996.

AUTOMATED HEALTH OPERATIONS FOR THE SAPPHIRE SPACECRAFT

Michael A. Swartwout[*] and Christopher A. Kitts[†]
Stanford Space Systems Development Laboratory
Stanford, CA 94305-4035

ABSTRACT

Stanford's Space Systems Development Laboratory is developing methods for automated spacecraft health operations. Such operations greatly reduce the need for ground-space communication links and full-time operators. However, new questions emerge about how to supply operators with the spacecraft information that is no longer available. One solution is to introduce a low-bandwidth health beacon and to develop new approaches in on-board summarization of health data for telemetering. This paper reviews the development of beacon operations and data summary, describes the implementation of beacon-based health management on board SAPPHIRE, and explains the mission operations response to health emergencies. Additional information is provided on the role of SSDL's academic partners in developing a worldwide network of beacon receiving stations.

KEY WORDS

Health management, Beacon, Spacecraft operations, Data summary, Anomaly detection, Automation.

INTRODUCTION

Two trends are forcing significant changes in spacecraft operations. The first is pressure to reduce cost, such as ground-to-satellite communication links, operator man-hours, and operator training. The second is the growth of multi-satellite, coordinated missions; not only are there more spacecraft, but such missions introduce systematic problems related to constellation management. Taken together, these changes make the operations problem intractable using standard operator-intensive methods. New approaches to spacecraft operations are necessary.

One proposed approach to handling these new constraints is automated health management. This method reduces cost in two ways: operator workload is reduced by performing routine health monitoring using

[*] Doctoral Candidate, Aeronautics & Astronautics, Stanford University
[†] Doctoral Candidate, Mechanical Engineering, Stanford University. Caelum Research Corporation, NASA Ames Research Center, CA.

automated systems; and communication requirements are reduced by migrating this automation to the spacecraft. Of course, spacecraft autonomy brings new challenges in determining the amount and type of information necessary to relay to ground operators. For example, operators will no longer be able to access an archived database to "catch up" on spacecraft health history. There are also questions of how robust spacecraft autonomy is to unexpected behavior and events. In meeting such challenges, the role of telemetry in spacecraft operations is being redefined.

After reviewing the laboratory's satellite and operations architecture, this paper outlines the beacon-based health management approach undertaken by the Space Systems Development Laboratory (SSDL) at Stanford University. The ASSET experimental mission operations system is being used to develop methods for automated health management, with the SAPPHIRE microsatellite as a testbed. After background information about the satellite and operations architecture, this paper outlines the health management system. Questions are raised about informing operators of important issues on-board, leading to a new view of telemetry. This paper also highlights the role of universities in researching and developing telemetry systems.

THE SPACE SYSTEM DEVELOPMENT LABORATORY RESEARCH PROGRAM

SSDL was chartered in 1994 to provide world class education and research in all aspects of spacecraft design, technology, and operation. To achieve this goal, SSDL members enroll in a comprehensive academic program composed of coursework, project experience and research investigations. As one of their investigations, SSDL is actively involved in research in spacecraft operations and automation.

The Satellite Quick Research Testbed (SQUIRT) Microsatellite Program - The SSDL SQUIRT program [1] is a yearly project through which students design and fabricate a real spacecraft capable of servicing low mass, low power, state-of-the-art research payloads. By limiting the design scope of these satellites, the project is simple and short enough so that students can see a full project life cycle and are able to technically understand the entire system. Typical design guidelines for these projects include using a highly modular bus weighing approximately 25 pounds, a hexagonal form that is roughly 9 inches high by 16 inches in diameter, amateur radio communications frequencies, and commercial off-the-shelf components. Missions are limited to about one year of on-orbit operation. Since little money is available for operations, a highly automated mission control architecture is being developed.

The Stanford Audiophonic Photographic Infrared Experiment (SAPPHIRE) Microsatellite - SAPPHIRE is the first SQUIRT spacecraft. Its primary mission is to characterize the on-orbit performance of a new generation of infrared horizon detectors, in addition to flying two student payloads (a digital camera and a voice synthesizer). Student research interests are also driving experiments in nontraditional sensing [2] and automated operations. SAPPHIRE is hexagonal, measuring 17" across its longest dimension and 13" high. It is primarily constructed of commercially available equipment, such as amateur radio kits and a Motorola 68000 series microcontroller, with some space-qualified elements, such as batteries. SAPPHIRE is being completed by a core of volunteers and research students, and is currently undergoing environmental testing.

The Automated Space System Experimental Testbed (ASSET) System - The ASSET system [3] is a global space operations network under development within SSDL. The first goal of this system is to enable low-cost and highly accessible mission operations for SQUIRT microsatellites as well as other university and amateur spacecraft. The second goal of this system is to serve as a comprehensive, low inertia, flexible, real-world validation testbed for new automated operations technologies. Figure 1 shows a high level view of the ASSET mission architecture. The basic components include the user interface, a control center, ground stations, communications links, and the target spacecraft. During the current developmental phase, a highly centralized operations strategy is being pursued with nearly all mission management decision making executed in the control center. These tasks include experimental specification, resource allocation throughout the ground and space segment, anomaly management, contact planning, data formatting and distribution, and executive control.

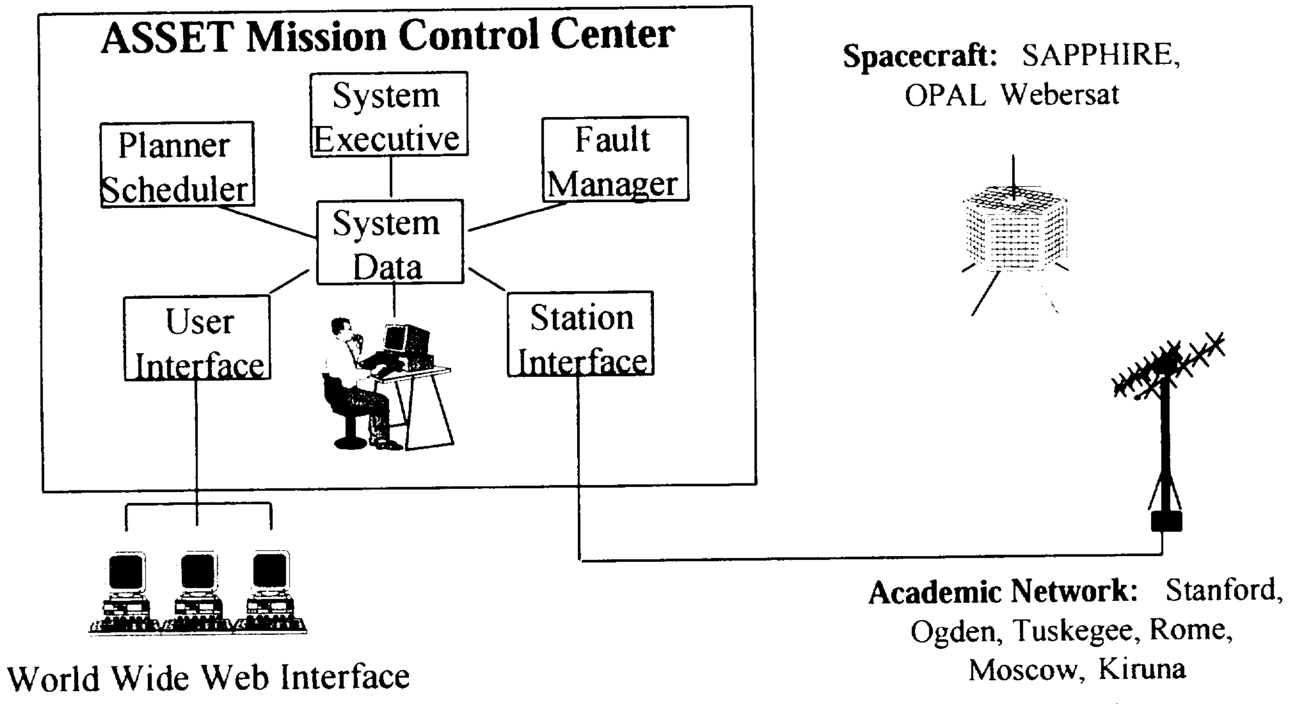

ASSET Mission Control Center
System Executive
Planner Scheduler
Fault Manager
System Data
User Interface
Station Interface
Spacecraft: SAPPHIRE, OPAL Webersat
Academic Network: Stanford, Ogden, Tuskegee, Rome, Moscow, Kiruna
World Wide Web Interface

Figure 1 *- The ASSET Space System Architecture*

SAPPHIRE and all future SQUIRT satellites will be operated through ASSET. In addition, controllers for a number of other university and amateur satellites have expressed in becoming part of the system. As for ground stations, the Ogden and Stanford ground stations are the first two facilities to be included. Several other stations throughout North America and Europe have been identified for future integration.

BEACON-BASED HEALTH MANAGEMENT DESCRIPTION

A beacon-based health management concept was proposed for a deep-space mission to the planet Pluto [4]. This concept is being prototyped as a part of the SAPPHIRE mission [5]; its main features are summarized, below, and the new elements are further detailed. The signal flow is presented in Figure 2.

SAPPHIRE Health Monitoring - SAPPHIRE will monitor its own telemetry sensors, comparing measured values with commandable entries in a state-dependent limit table. Certain measurands will be validated by aggregate analysis. For example, all body-mounted solar panels cannot see the Sun at once; this knowledge is used to help verify that the solar panel current readings "make sense". These modest steps provide SAPPHIRE with an anomaly detection system far more mature than most spacecraft.

Depending on the seriousness of the limit violation, the spacecraft health is assessed to be one of four values. For example, when measurands are within limits, the spacecraft is judged to be Normal. Out-of-limits with moderate impact, such as an overheating transmitter, is considered an Alert. Out-of-limits that can rapidly jeopardize the mission elevates the health status to Critical. Finally, Emergency condition is defined to be an unexplained computer reset. Note that the rules by which measurands trigger the modes, and the limits for each, are defined by the operations team. This is because the four beacon modes are intended to be a mapping from spacecraft state to operator action.

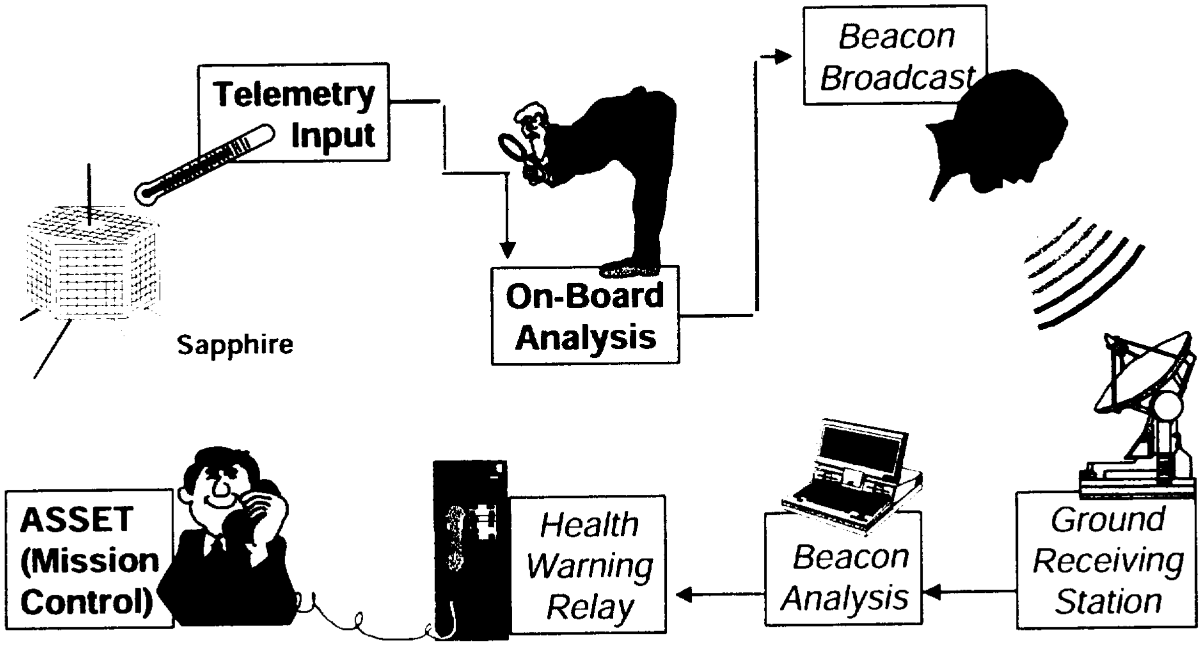


***Figure 2** - SAPPHIRE Health Beacon Signal Flow*

Health Beacon Transmission - The health beacon is transmitted with the AX.25 packet format common to Amateur radio users. SAPPHIRE's main transmitter also serves to broadcast the beacon message, with an output power of around 500mW. The message consists of SAPPHIRE's call sign (part of the protocol) and the health assessment identifier. In this manner, the information from thirty-two analog sensors and six digital lines has been compacted into a few bits that tell an operator whether or not SAPPHIRE can continue to perform its mission. And while such information once had to be collected over time for eventual download and processing at mission control, spacecraft health is now continuously monitored and available anytime the spacecraft is within range of a low-cost receiving station.

Receiving Station - SSDL has partnerships with universities in Alabama, Montana, and Sweden to develop a simple receive-only system for health monitoring. These stations will listen for SAPPHIRE beacon transmissions and notify mission control of the results by electronic mail. It is intended to put these stations at locations around the world, giving SAPPHIRE (and other spacecraft in the ASSET network) near-global coverage for health monitoring.

Mission Control Center - Once mission control receives a beacon monitoring update from a remote station, it logs this information and then takes appropriate action. Depending on the health assessment, there are varied responses, from storing the update in the system database to paging the operator on call and rescheduling the network to contact and recover a failed satellite.

Implementation - Currently, SAPPHIRE's flight software is being modified to include health monitoring and beacon message broadcasts. Beacon receiving stations are being developed by the university partners. Automatic notification from a beacon monitoring update has been demonstrated, and ASSET's scheduling capabilities are still in development. A functional demonstration of the complete system will take place in the fall of 1997.

ENGINEERING DATA SUMMARY

The second challenge in automated health management is to provide the operators with the context necessary to make informed decisions. Normally, these operators spend hours closely watching the various spacecraft measurands, enabling them to spot trends and develop an intuitive feel for the various subsystems. Familiarity with the system is extremely important for such tasks as anomaly management, because not all faults are directly observable by measurands, nor is it always clear which recovery action will least impact other components. However, in this approach to automated health management, the spacecraft is now performing the routine data analysis once handled by operators. In addition, the raw telemetry is no longer available on the ground for offline analysis or operator perusal.

The problem statement for data summary, then, is to reduce the set of on-board sensor data to what is necessary for operators to carry out their duties. This reduced telemetry set is the only information sent back to mission control, apart from science or mission data from the payload. Sending additional information to the ground wastes scarce communication resources; sending less prevents operators from ensuring mission success.

Now, this is not simply an application for data compression, for the goal is not use the summary to reconstruct the complete measurand set. Rather, it is to identify the information valuable to operators and present it succinctly. Numerical compression makes no use of the built-up body of knowledge about a spacecraft. Such knowledge aids the transformation of measurands to into data summaries that are full of meaningful content. Such an approach promises greater data savings over compression and provides operators a head start in such tasks as anomaly management.

The task of anomaly management was chosen as the first candidate for data summary. Once an anomaly is detected, operators must determine what fault there is, if any, and take action to return the spacecraft to as normal a condition as possible. Two candidate solutions have been identified. Both share their roots in a concept of model-based analysis, taking advantage of what is known about the spacecraft components and their interactions in developing summaries. Other, non-model-based approaches exist, such as statistical summaries and curve-fitting, but have not been selected as initial candidates.

The first candidate approach is to log only the out-of-limits measurands. The limits will be defined according to component and system reliability values, and will vary for different operational modes. This is expected to reduce the amount of data sent back by more than a factor of ten[‡]. The shortcoming of this method is that all the data leading up to the anomaly is ignored, so it is difficult to perform trend analysis. Secondly, this method ignores other components that are exhibiting unusual, but not out-of-limits, behavior, which may give hints as to the source of a hidden fault.

The next candidate approach integrates the work done for advanced on-board anomaly detection using artificial intelligence techniques. Such a system not only monitors the on-board sensors, but it analyzes the results to identify faults, using a built-in spacecraft model. In contrast to the first method, the measurand limits are dynamically set models of each component. The advantage to this method is that the same model can be reproduced, which means that the spacecraft needs only to send back the information to update the ground model, such as inputs and the functional status of components. Should an anomaly occur, the spacecraft is responsible for determining the source of the anomaly; this information is considered an update to the spacecraft state and the ground is notified only if this requires a change in operations. It is unclear what the savings in transmitted data will be, though expectations are that this method should out-perform the first candidate while further reducing the responsibilities of the operators. A known drawback is that it requires significant on-board computation, as well as a detailed model of the whole spacecraft. Additionally, it is unclear whether all this computational effort succeeds in paring the data down to that which the operators require. Further study is needed.

THE UNIVERSITY'S ROLE IN TELEMETRY RESEARCH

SSDL's work in automated health management provides insight and examples in how universities can contribute to the research and development of telemetry approaches. Put bluntly, universities have the opportunity to fail – for failure is a fundamental part of the education process. A university laboratory like SSDL can afford to investigate risky projects or to approach problems from widely different angles. This is because the primary outputs of a university are not successful products or methods, but well-trained, capable students. Obviously, understanding why a specific approach did not work is critical to the development of good students. But a failed project is not a true failure if new solutions (and better-informed students) come out of it. Unlike a competitive industrial project, universities can afford to fail.

[‡] An Air Force study [6] determined that the spacecraft was determined to be "normal" in more than 90% of contacts. Admittedly, such decisions involved an operator who could ignore the many false alarms generated by inadequate limit definitions. But this new approach assumes a mode-dependent limit checking, adjustable over time to account for component degradation, which should eliminate most of these false alarms.

FUTURE WORK

Obviously, the highest priority is to complete the development of the beacon-based health monitoring system. The summer will be spent completing and testing SAPPHIRE's flight software, including the on-board health management. Work will continue in coordinating the construction of beacon receiving stations from the detailed designs that have been generated. And ASSET's capabilities will be expanded to allow for operation of SAPPHIRE through its system, culminating in a complete system demonstration in fall 1997.

Once those elements are in place, the value of this automated health monitoring approach can be demonstrated. SAPPHIRE will be operated for a period of a week or more, first using the standard operator-intensive approach, then with the beacon-based monitoring. Three factors will be compared: accuracy of fault detection, speed of response to on-board anomalies, and the cost in operator hours and active satellite-to-ground contacts. Since the automated fault detection methods are identical to the ones used by operators, it is expected that short-term accuracy will remain the same. However, significant improvements are expected in both speed of response and cost, since this passive network will continuously listen over a wide area and operators are required only for anomalous events.

Engineering data summary research is a wide-open area that needs further investigation. The candidate approaches need to be evaluated for performance, and more candidates need to be developed. While data summary will not be a part of SAPPHIRE's flight software, summary algorithms will be implemented on the ground to assess their impact on flight data analysis.

CONCLUSION

The use of a low-power beacon and distributed listening stations is a viable solution to the new challenges in spacecraft health management. Automation of limit detection and migration of this process to the spacecraft will significantly reduce the spacecraft-to-ground communication link and the time spent by operators. This beacon-based health management approach will be demonstrated using the SAPPHIRE spacecraft through the ASSET operations architecture, in cooperation with Stanford's university partners throughout the world.

Given this anomaly detection approach, a primary concern is to assist operators given their inability to access the complete measurand set. Two candidates in the area of anomaly management have been defined: one summarizes all out-of-limits measurements, the other uses advanced fault detection methods to reduce the measurands to an abstraction of the spacecraft operational status and the known inputs. Neither approach has proven to be completely satisfactory, and both approaches will be further refined to develop new candidates.

Work on this research topic highlights the need for better defining the information – both the type and amount – needed for spacecraft operators to perform anomaly management. Answers to that question will also affect the future role of telemetering systems in spacecraft operations. The traditional function of spacecraft telemetering – measuring sensors at a distance – is being challenged by the high cost of remote operations. Data interpretation in the next generation of spacecraft be handled on-board, with only an abstraction of the measurands returned to the ground. Such a profound change in telemetering deserves further study. The university environment, with its greater flexibility and room for failure, is an important contributor to such investigations.

ACKNOWLEDGEMENTS

The authors wish to thank of the SAPPHIRE team for their dedication and efforts, especially the flight software programmers, Rajesh Batra and Kenneth Koller. The Jet Propulsion Laboratory and NASA Ames Research Center are acknowledged for their support and assistance with this research project. Finally, appreciation is extended to all of the institutions and organizations that have participated in, contributed to, and provided feedback concerning various aspects of the ASSET system. This includes all of the previously mentioned schools that are interested in integrating their spacecraft and/or ground stations with the ASSET network. Special thanks goes to the receiving station partners at Tuskegee Montana, and Sweden. This work has been performed in partial satisfaction of graduate studies at Stanford University.

REFERENCES

[1] Kitts, Christopher A., and Robert J. Twiggs, "The Satellite Quick Research Testbed (SQUIRT) Program," Proceedings of the 8th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 16-22, 1994.

[2] Swartwout, Michael A., Tanya A. Olsen, and Christopher A. Kitts, "The Omni-Directional Differential Sun Sensor", Proceedings of the 31st Annual International Telemetering Conference, Volume XXXI, IFT, Las Vegas, NV, 1995, pp. 730-736.

[3] Kitts, Christopher A., "A Global Spacecraft Control Network for Spacecraft Autonomy Research." Proceedings of SpaceOps '96: The Fourth International Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, September 16-20, 1996.

[4] Wyatt, E. Jay, and John B. Carraway, "Beacon Monitoring Approach to Spacecraft Operations", Reducing the Cost of Spacecraft Ground Systems and Operations, Oxfordshire, United Kingdom, September 27-29, 1995.

[5] Swartwout, Michael A., and Christopher A. Kitts, "A Beacon Monitoring System for Automated Fault Management Operations", Proceedings of the Tenth Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 16-19, 1996.

[6] Farmer, Mike and Randy Culver, "The Challenges of Low-Cost, Automated Satellite Operations", Proceedings of the 31st Annual International Telemetering Conference, Volume XXXI, IFT, Las Vegas, NV, 1995, pp. 203-209.

Engineering Data Summaries for Space Missions

Michael A. Swartwout
Space Systems Development Laboratory
Durand 250
496 Lomita Mall
Stanford, CA 94305-4035
650/725-6794 (rcktboy@leland.stanford.edu)

Abstract— New paradigms in spacecraft design are leading to radical changes in spacecraft operations. Increased constraints on resource usage and greater focus on operations costs require new approaches. One such method, beacon-based health monitoring, automates the task of routine health monitoring and migrates the process from the ground to the spacecraft.

The performance of this automated method is further improved by a supplemental approach that monitors the long-term health of the spacecraft. Called "engineering data summarization", this process has the responsibility of creating an on-board summary of the spacecraft state of health, tracking notable sensor values and trends as appropriate. Every few weeks, the summary is transmitted to ground operators. The purpose of the summary is to provide operators with context about the spacecraft's state.

Building on the systems for spacecraft operations being developed at Stanford's Space Systems Development Laboratory, this paper is the first step in developing a methodical study of engineering data summarization. Five simple solutions are proposed, and each is examined using newly-established, competitive metrics. Analysis of the solutions' characteristics leads to a definition of the problem's "solution space." These results point to the next steps needed for a thorough characterization of the engineering data summarization problem.

TABLE OF CONTENTS

1. INTRODUCTION

Two pressures – performance and cost – are pushing a revolution in methods of spacecraft mission operations. The advent of large, interdependent constellations like global satellite communication networks increases the scope of operations by an order of magnitude, creating system-wide effects not present in single-spacecraft missions. The scale and nature of these effects make traditional operator-intensive solutions intractable. New constraints on the Deep Space Network greatly increase the cost of communicating with deep-space vehicles, especially during the "idle" years of cruise phase. For these and other situations, the need is not merely to lower costs to make the mission more competitive; methods must be developed to enable the mission to be accomplished at all.

One proposed approach to improve spacecraft operations is beacon-based health monitoring, also called "beacon monitoring." Since more than 90% of all health assessment contacts do not require any response by the operators [1], automating the process of anomaly detection can significantly reduce cost. In this method, the spacecraft analyzes its own sensor data to assess its state of health; this abstracted state is broadcast in the form of a low-power, low-bandwidth beacon. Human operators are involved only to respond to abnormal conditions; the state-of-health beacon assists by indicating the neeeded changes in operational procedures.

There are, however, some long-term drawbacks to this approach. One of the primary goals of beacon monitoring is to reduce the amount of data sent to the ground, which is achieved by eliminating the download of telemetry data. But that telemetry set is used in other tasks as well. Operators gain intuition about the performance and characteristics of each spacecraft and component by examining the real-time telemetry and through simulations with the exhaustive data archive. They develop informal heuristics for troubleshooting a spacecraft and have learned to distinguish "quirks" from malfunctions. If the operators do not have some means of developing detailed understanding about the spacecraft, they cannot adequately fulfill their duties.

Therefore, in order to fully obtain the benefits of the beacon monitoring method, the "fast loop" of real-time health assessment must be supplemented by a "slow loop" to study the long-term behavior of the spacecraft. Moreover, there are operator tasks other than health management which would benefit from added information about the vehicle. Examples of these other responsibilities are command verification and understanding third-party payload usage.

The supplement to beacon-based health monitoring is generically called "engineering data summarization", or "Summary." This term is intended to encompass the family of implementations whereby the spacecraft creates a second set of abstractions about the sensor telemetry; this information is sent back to the ground to provide context for operators. The form of the data, the amount, and the frequency of summary downloads are all variables to be optimized, resulting in the least cost – in terms of communication bandwidth and operator effort – while maintaining the performance margins of operator-intensive missions.

This paper will outline the concept of engineering data summarization. It is intended to be the first step in the development of a methodology for long-term automation of spacecraft mission operations. Section 2 will define the problem, emphasizing the initial assumptions and constraints used in this scope. Particular attention will be paid to clearly defining the roles and relationships between beacon-based health monitoring and engineering data summarization. In Section 3, five strawman Summary solutions are proposed; the purpose of these solutions is to identify important issues in the Summary problem. Analysis of these solutions helps develop the performance metrics as described in Section 4. These assessments of effective solutions are based on competitive measures of quality, cost, and timeliness. In Section 5, study of the strawman approaches leads to the definition of the solution space. Section 6 provides conclusions of the work thus far and describes the next steps for developing the methodology.

2. Problem Statement

Because engineering data summarization is a fairly recent concept [2], the scope and nature of the problem is still not fully defined. Therefore, it is important to describe both what Summary is and what it is not. Specifically, engineering data summarization must be distinguished from beacon-based health monitoring.

Beacon-Based Health Monitoring

Beacon monitoring is, essentially, two migrations of the spacecraft anomaly detection task: the responsibility for routine health monitoring shifts from an operator to an automated process, and this process is moved from the ground to the vehicle. The mission saves operator man-hours and communications bandwidth because of each respective migration. Where there was once a continual data stream requiring teams of operators to analyze, beacon monitoring creates a small (on the order of a few bits) signal that requires operator attention only in the event of an anomaly.

Motivated by the opportunity to cut the cost of operations, beacon-based health monitoring is being studied for several projects. It is an essential element of JPL's proposed new Pluto/Europa/Sun missions [2], a technology demonstration for the New Millenium Program's Deep Space 1 mission, and has been proposed for the Air Force Satellite Control Network [3]. In 1998, Stanford University's SAPPHIRE microsatellite [4] will conduct validation experiments of this concept for JPL [5, 6].

These projects have also determined that operators need more information about the vehicle than what the small, simple health assessment flag can provide. It is not a matter of recording the full telemetry sets once an anomaly is detected, though that may be an important part of fault isolation and recovery. Instead, operators need additional information about the state of the spacecraft *before* the anomaly, so that they can perform all of their tasks. This additional information is the Summary.

Table 1 describes the differences between beacon-based health monitoring and the Summary. While both are essentially operational tools that convert on-board data into information useable by operators – in fact, one could argue that beacon monitoring is nothing more than a specific implementation of the Summary – other factors associated with implementation make them worth distinguishing. For example, the primary task of beacon monitoring is to fulfill the responsibility of health monitoring by providing an instantaneous indication of a need for a change in operations. By contrast, the Summary is intended to enhance the performance of the operators (and therefore the vehicle) within any given mode. A useful but limited analogy is to liken the Beacon to the function of health monitoring and the Summary to the function of examining the telemetry archive. Each utilizes the vehicle data for related, yet distinct, purposes.

Table 1 Comparison of Beacon and Summary

Element	**Beacon Monitoring**	**Engineering Data Summarization**
Time	Fast	Slow
Output Size	Very Small	(Undetermined)
Input	Telemetry	Telemetry
Assessment	Abstracted State	Abstracted States Processed Data Raw Data
Role in Operations	Indicates a Change in Operational Modes	Assists Effective Performance of Operators and Vehicle within a Mode
Analogous Function	Health Monitoring	Consulting Telemetry Archive

The purpose of this study is to develop a methodology for engineering data summarization in space missions. Research in beacon-based health monitoring is further discussed in the aforementioned references. While beacon monitoring is an important precursor to an effective Summary, the subjects are distinct enough to allow for separate investigation.

Example Scenarios

For a definition of the engineering data summarization problem, it is helpful to create two examples, drawn from two general classes of space missions. The first is an Earth-orbiting constellation of spacecraft, which involves issues of large-scale, complex systems, rapid response times, and high-performance payloads. The second is a single deep-space vehicle, which involves issues of long communications delays and robust performance. Between these two examples, most of the crucial needs of data summarization can be identified.

Scenario 1: Global constellation - Assume that a space constellation of several dozen communications satellites has implemented beacon monitoring. Given the usual variations in manufacturing quality and the complexity of these spacecraft, it is not surprising that the heat pipes for the batteries of Vehicle 28 perform slightly worse than expected, and this variation was not detected before launch. Thus, during the season when the orbital plane brings more sunlight onto that region of the spacecraft, the batteries are slightly hotter than average. The difference is small - certainly below the limits defined for an abnormal condition - and thus is not detected on-orbit, either.

When the nearby power regulator starts registering dangerously high temperatures, the beacon system detects the anomaly. The operator called to investigate has never directly operated this spacecraft before, and because the quirky heat pipes have not been identified, she may believe that the regulator is affecting battery temperature. This misunderstanding could lead to unneeded changes in operations to protect the battery, and at best confuses the task of isolating the heat effects. The operator would be greatly aided by the ability to know that the battery quirk is a long-standing phenomena and is probably not related to the overheated regulator. In other words, the chances for mission success would be greatly enhanced by providing some means of reproducing the long-term trend analysis (as used in "typical" operations of today) on the ground.

Scenario 2: Deep-Space Mission -Pluto Express is in its seventh year of a ten-year cruise to the outermost planet. Everyone originally involved in the development and check-out of the spacecraft has retired or been reassigned. The beacon-based health monitoring registers an alarm, and the operator on call discovers that one of the propulsion tanks has unusually low pressure. Since a leaking tank would require profound changes to future operations, it is imperative to determine if this is a true leak or a problem with the sensor. The operator would be greatly assisted in his tasks by the ability to look at the past history of the sensor, especially during maneuvers where general sensor performance could be predicted, and over the previous few days to identify performances characteristic of a leak.

Granted, these two examples are simplistic and the problems addressed could possibly be solved by better operator training, redundant sensors and spacecraft check-out procedures. But that is precisely the point: long-term functionality of a mission using beacon-based health monitoring requires additional operational procedures and possibly changes to the spacecraft architecture. Automated health monitoring alone cannot account for long-term vehicle health.

Still, it is unlikely that simple adjustments to operational procedures will ensure adequate performance. For example, putting tighter bounds on limit-checking, to catch the small discrepancies, leads to a higher false alarm rate with commensurately higher operating costs. Accurate, detailed models require additional efforts to maintain and update and are susceptible to unmodeled or unobservable inputs. Additional sensors cost mass, power, computational ability and operator effort. Regular telemetry downloads increases the cost of communication bandwidth.

While changes such as those listed above may indeed be elements of the best low-cost mission operations scenario, it is imperative to establish a careful methodology. Current trends in spacecraft operations laud the benefits of automation, but automation requires clear methods for proper implementation. The spacecraft operations business has, over the last forty years, invented many "rules of thumb" and informal heuristics. These heuristics often do not translate well into automated methods. In order to create effective automated solutions, these informal ideas must be formalized.

Assumptions

Because this is only a preliminary investigation into the nature of the engineering data summarization problem, or Summary Problem, it is helpful to reduce the study's scope. Several key assumptions will assist in creating a topic that, while not complete, exhibits the most important characteristics of the real-world scenarios.

It is assumed that beacon-based health monitoring has been included in the spacecraft specifications and design, fulfilling the functions defined earlier in this section. And while it is expected that the exact parameters of this beacon system would be optimized to fit the specific mission, some basic characteristics can be assumed.

(1) The beacon monitoring system performs automated health monitoring of the spacecraft, informing operators in a health state change that requires an operational procedures change.
(2) The beacon monitoring system is not required to perform more advanced autonomous functions, such as fault isolation and correction. Such functions may be incorporated into more advanced Summary techniques.
(3) The engineering data summarization element of the spacecraft and the beacon-based health monitoring system have access to the same reasoning and analysis capabilities.

The point of assumption (3) is to emphasize the existence of strong coupling in functions - and therefore designs - between the beacon monitoring and Summary systems. As will become evident, a vehicle using enhanced methods in health monitoring should also apply those methods to the Summary; similarly, a highly-capable Summary solution will allow for a more capable Beacon system.

(4) Human operators will be involved in the more complicated and/or unexpected functions of spacecraft operations.

Whether or not operators are involved – and to what degree they are involved – greatly affects the type of Summary solution created. The presence of human operators is one of the driving factors behind the Summary. Of course, it may eventually be feasible to completely eliminate humans from health monitoring and other operations functions. But given the reliability and capabilities of present-day space systems, it is more reasonable to assume that automated systems will face problems they are unable to handle, requiring the intervention of operators.

(5) The role of operators is limited to that of health management.

This assumption helps to clarify (4), and limits the scope of the study to the following operator tasks: identifying anomalies, isolating their sources, assessing the impact of the anomaly, taking action to recover from faults, and altering the understanding of the system based on the new information. The Summary does not perform these tasks, but provides the operator with information that was once directly available from vehicle telemetry.

(6) The communications equipment used to transmit the Summary will be no more capable than that used by "normal" spacecraft for telemetry downlinks.

This assumption emphasizes that the purpose of this study is not to rely on new technologies to send more information in less time, but to discover systems that enable operators to do more with less information. Mission designers will always push the communication capabilities to their limits; rather than seeking ways to create and manipulate more data, it is important to seek ways to take advantage of existing but underutilized knowledge of the system.

The Summary Problem

The general statement of the Summary Problem is this: **Engineering data summarization is to provide the necessary information for operators to carry out their tasks, while minimizing both the efforts of these operators and communications resources.** Granted, this statement is quite vague on what constitutes "necessary information," and definition of "minimal" is equally nebulous. The study of metrics in Section 4 will explore the latter subject; the rest of this section explains the former.

As highlighted in the examples, the operators need to be provided with context and history about the vehicle. However, they must glean this context from a reduced – or "summarized" – set of information. From the standpoint of contingency operations, therefore, the fundamental goal of engineering summarization is to be able to reproduce information about the vehicle for use by operators while improving performance metrics such as communications cost and operator efforts.

There are many methods to accomplish this goal; determining which method is "best" is done using the metrics described in a later section. Note that a Summary solution need not recreate the *full* telemetry set from a reduced supply of information; it must provide that information which operators need to fulfill their responsibilities. While effective solutions may indeed involve the ability to wholly or partially rebuild sensor data history, this is not an expected.

3. Initial Solutions

As an aid in understanding the nature of the solution space, and in order to provide starting points for developing solutions to the summary problem, five strawman solutions have been proposed. Each places exaggerated emphasis on one or more characteristics; the intent is to identify how each of the parameters affects the performance of the overall solution. Again, the emphasis of this study is not to propose viable Summary solutions, but to better understand the nature of the problem itself. The characteristics of each candidate are presented in Table 2, along with how they perform for the space mission examples.

Reduced Sample Rate

One of the simplest methods of summarization is to downlink a partial telemetry set. If the ground-based communications system has a maximum data downlink capability, then the solution is to prioritize the sensor information and transmit the maximum amount. The duration between transmits is altered to fall just within the specifications. The system adds no additional modeling or sensors beyond that which would normally be on board. The information saved for download would be what is considered most important: short bursts around interesting events and then snapshots spaced out to fill in the rest of the available time.

This approach is very simple to implement. However, one shortcoming is that it requires advance understanding of the spacecraft and an expectation of what are likely to be the most important events. Moreover, it does not reduce the cost or training of operators; this approach assumes they will continue to familiarize themselves with the reduced data set as was done with the complete set. And, this method does not consider that important information may be lost during the no-sample periods.

For example, in the Earth-orbiting constellation, operators could designate the basic kinds of information about each vehicle that helps them to perform their tasks. All vehicle Summaries could be configured to emphasize the relevant parameters in their download. It is not at all clear, however, that the warm battery problem of the example would be spotted in this selective Summary process – unless battery temperatures were already known to be unusually important.

Still, this approach highlights the fact that sensors have varied importance for different phases of operations. Some sensors convey more critical information than others, and some events where many items are changing at once, such

Table 2 Use of Initial Summary Solutions for Example Problems

Solution	Deep Space	Earth constellation
Reduced Sample Rate	Helps prioritize data Requires significant bandwidth May "lose" pressure sensor trend	Helps identify system-wide issues Requires significant bandwidth May "lose" battery history
On-Board Database	Hampered by long delays Helped by "Anticipating Summary" Requires very robust memory "Finds" pressure sensor trend	Helped by short transmit times Requires robust memory "Finds" battery history
Statistical Summary	Hampered by small "sample size" Relies on expectations May "lose" pressure sensor trend	Helped by large "sample size" "Finds" battery history
Alarm Threshold Summary	Very low bandwidth Requires accurate thresholds "Loses" pressure sensor trend	Very low bandwidth Requires accurate thresholds "Loses" battery history
Model-Following Summary	Helped by "stable" models Helped by very low bandwidth "Finds" pressure sensor trend	Hampered by many varying models "Finds" battery history

as a thruster firing, may require more study than a week of dormant cruise. Effective summary solutions will assess the relative worth of spacecraft sensors in order to select the most essential information.

On-Board Database

Another straightforward approach is to store all telemetry data on board the spacecraft. The telemetry archive is not lost; it is maintained in spacecraft memory instead of on the ground. The operators can request any information they want, but otherwise no data is ever sent to Earth.

While at first this approach seems ludicrous because it requires immeasurable amounts of on-board memory and assumes that operators will have the time to retrieve desired data, it does have some helpful ideas. It emphasizes the usefulness of contingency-generated Summaries that pay close attention to component(s) with anomalous conditions. But instead of waiting for a user request, the vehicle should anticipate the demand for additional information about these suspect parts, search its memory for related information and take additional samples to clarify the anomaly. It would thus prepare an initial report, with background information, for use by the operators in troubleshooting.

This candidate seems to be more appropriate for an Earth-orbiting mission where a spacecraft database can be rapidly and repeatedly queried. When the transmit time can be on the order of hours, though, it makes little sense to send requests for information. In the "anticipating Summary" case, it would be very useful for the deep-space vehicle to search for relevant information to download while the ground team is being assembled. If the pressure sensor was giving an anomalous reading, the Summary could call up its time-history and prepare it for download.

Statistical Summary

A slightly more involved approach is to perform statistical analyses on the sensor outputs. The time-history of sensor data can be examined for such items as trends, frequency spectra, and averages. In order to make use of this information, it is necessary to have expected performances of these sensors. However, the generation of expectations and comparisons can all be performed on the ground, where the baselines can be easily updated as new information becomes available. This method emphasizes the usefulness of expectations; understanding about the vehicle leads to predictions about its behavior, which can be used to identify both quirks and faults.

While promising to be a capable solution, using techniques that are well-understood in the scientific and engineering communities, this approach does not take advantage of the full information available. The relationships between components are completely ignored. For both the Earth-orbiting and deep-space missions, the use of statistical summaries might catch the battery and pressure sensor problems – but, especially in the case of the pressure sensor, actually making use of its statistics requires additional understanding about the vehicle in order to create reasonable expectations. On the other hand, in a constellation there are many spacecraft, which automatically increases the "sample size" for the statistics.

While such information may not be necessary for every effective Summary, it is worth investigating more model-based approaches to see if additional performance benefits or cost savings are possible. (Alternately, a statistical-based method that uses causal models, such as the one developed by Doyle [7], could be used. Basic information about how one component affects another is assembled into a relationship tree; statistics are kept on the responses of each sensor and this information is used to predict which components are behaving properly.)

Alarm Threshold Summary

Since most modern spacecraft already employ alarm thresholds for health monitoring, it makes sense to explore this technique for the Summary. A basic model of the spacecraft is created, with state- and mode-based abnormal and emergency alarm thresholds defined. Those sensor outputs which fall outside the normal limits would be stored for Summary download. Great savings in bandwidth would result, since only the "abnormal" information is kept. Like the statistics approach, the primary shortcoming in this method is the loss of valuable information about the relationship between components. One component within normal limits may point to a problem with a component that is out of limits, but the former data is ignored. In fact, for both examples, the vital information about the components would be "lost" because it was not in the abnormal range.

The lessons learned from this method are the importance of saving information that may seem to be "normal", and also the emphasis on using the already-assembled body of knowledge to simplify the approach.

Model-Following Summary

A more complex approach that promises great reduction in downloaded data is a model-based summary. Detailed input-output transfer functions are kept for every component of interest; identical models exist in the vehicle computer and on the ground. Given the inputs to the system, very precise outputs for each component are generated, and this information is compared to the sensor data. Only those points that reflect significant deviation from the model are stored, along with an the system configuration and inputs at the time of the discrepancy. Thus, the deviations from expected behavior and the context of the situation are explicitly identified and stored. Using this information, the ground-based model can faithfully simulate the performance of any component on board the spacecraft.

For a deep-space mission, especially during cruise when the external environment and vehicle modes are changing quite slowly, this approach has special appeal. The models of the spacecraft and its environment are stable over the long term, allowing for detailed simulations to be developed. The pressure sensor problem would be readily diagnosed, since its behavior would have been very closely followed. And the Summary need only pay attention to the discrepancies in the model – a relatively small amount of data to store for download.

The limitations of this system are related to the complexity of the modeling approach. Inaccurate models or unobservable conditions lead to holes in the summary, and effort is required to maintain the models as they change over time. Also, the ability to recreate the entire telemetry set probably indicates that excessive, unusable data will be downloaded.

4. PERFORMANCE METRICS

As demonstrated by the examples, most of the strawman methods are viable Summary techniques. With a little work, some sort of reasonable solution could be achieved. However, some options seem to be more appropriate, more effective, than others. A methodical approach to choosing the most effective solution requires the introduction of performance metrics.

Performance metrics should reflect the overall goals of the Summary Problem. Additionally, they should favor the solutions that are "competitive" in engineering practice [8], that is, those that increase the quality of the product, lower the overall cost, and reduce the time needed to produce the product. Summary metrics are presented in Table 3. Not surprisingly, these sets of goals are often in conflict: increase in quality comes at an added cost. How to choose between metrics is discussed following the metric descriptions.

Quality Measures

The "quality measure" is a reflection of the usefulness of the Summary. The fundamental issue of quality is whether or not an operator is able to acquire the information he needs. Determining a quantitative measure for total summary quality has proved elusive; at the least, it can be measured by comparing the performance of anomaly management tasks using the complete data, and then using the Summary. High-quality Summaries will duplicate the results of procedures performed with the original telemetry.

A related, though indirect, measure of quality is the effort required to use the summary. Since the man-hours spent on operations is more appropriate as a measure of *cost*, an applicable *quality* measure is the level of training. High-quality Summary solutions would not require extensive operator training. An admittedly inadequate means to quantify the level of training is the operator's salary, since that indirectly reflects his or her training and background.

A third quality measure is how effectively the system responds to anomalies. Solutions that perform well will reduce or eliminate the loss of payload operations due to uncorrected faults, because the summary helps the operator to efficiently recover the vehicle. One way of measuring this response is to keep track of the lost opportunities to perform payload operations because the spacecraft is not available. The Summary approach that results in the least vehicle "down time" is of highest quality.

As a whole, the quality measures apply similarly to both classes of Summary problems. Constellations want to reduce operator training because of the number of operators and spacecraft involved; deep space missions cannot afford to retain skilled operators during the years in cruise phase. For deep-space programs, down time relates completely to the phase of the mission, because obviously the loss of performance during a flyby is much greater than during cruise. A commercial Earth-orbiter, by contrast, relies on high-performance payloads to turn a profit; down time is extremely costly.

Time Measures

In a product development context, the "time measure" is an indication of how quickly the product can be delivered to the market; in the "faster, cheaper, better" spacecraft

Table 3 Engineering Data Summarization Performance Metrics

Type	Description	Metric	Measurements	Deep Space	Earth-Orbiting
Quality	Usefulness	*Operator Confidence*	*???*	Vital	Vital
	Operator Effort	Operator Training	*Salary*	Important	Important
	Down Time	Spacecraft Down Time	Lost bits/sec	Conditionally Important	Vital
Time	Time to Implement	Time to Check-Out	Man-hours	Desirable	Important
Cost	Operator Resources	Operator Man-Hours	Average hours/week	Vital	Important
	Communication Resources	Length of Contact Data Downloaded Time Between Contacts	Carraway and Squibb's Metric	Vital	Important
	Spacecraft Resources	On-Board Storage On-Board Processing Sensor Requirements	Bytes Bits/sec Price, Mass, Power, Size	Desirable	Desirable
	Ground Resources	Cost of Equipment	Dollars	Unimportant	Important

context, the time measure is an indication of how quickly the Summary service is available, that is, how quickly the Summary system could be implemented and functional in a mission operations concept. This is measured in the time taken to develop models, perform simulations, and carry out initial orbit checkouts to confirm that the system is working properly. Obviously, those systems which can be rapidly implemented and validated have better time measures than those that cannot. Missions with long checkout and/or cruise phases are less concerned with time measures, compared to a commercial payload.

Cost Measures

The two major "cost measures" involved in the Summary Problem are the time it takes for operators to perform their tasks, and the communications resources that are utilized. The former is rather straightforward to measure; the average total number of man-hours per week the operators devote to health management tasks indicates the effort required. This metric is particularly useful since managing the cost of operations was a fundamental motivation for establishing beacon monitoring, and thus Summaries. It is important for constellations to control the number and cost of operators. It is extremely important for a deep-space mission to slash the cost of operating the spacecraft during the long idle phases.

The communications resources cost measure is more difficult to quantify, since it includes several factors. The true cost of using a communications network is hard to determine since the systems are often shared between many programs and involve both fixed and recurring costs. However, the total cost of communications is a reflection of the amount of data being received, the duration of the contact, and how often the contacts occur. Since all three of these elements are easy to measure, a metric composed of them would be a good indication of communications cost. It is assumed that other factors – such as payload data – will be the drivers in defining communications hardware; the Summary will use whatever is already available to the vehicle. This assumption is a driving factor in developing deep-space Summaries, since communication involves long delay times and expensive equipment. It is less important, though still important, to an Earth-orbiting constellation that wishes to conserve its resources.

Carraway and Squibb [9] have proposed a set of metrics to indicate the level of autonomy for spacecraft. One metric in particular, the Spacecraft Engineering Analysis, provides an effective means of combining the above cost measures:

$$\begin{array}{c}\textit{Spacecraft}\\ \textit{Engineering Analysis}\end{array} = \left(\frac{NT}{DWN \times WORK}\right) \times \left(\frac{NT}{T}\right) \quad (1)$$

Where T = duration of track (hours)
NT = time between tracks (hours)
DWN = data downloaded (bits)
WORK = operator effort (man-hours)

This metric promises to be quite useful, since it penalizes large downloads, operator effort, and the duration of contacts. It also doubly rewards long no-contact periods, which reflects the overhead involved with scheduling and pre-calibration for a contact. In other words a few, long-duration communication passes are better than several shorter ones.

A third cost measure reflects the requirements a Summary technique levies on other vehicle subsystems. This question is somewhat harder to quantify, since Summary will often be an enabling technology – meaning that the mission *must* perform a Summary or the mission cannot be accomplished. If a particular solution requires extensive computing power, then that becomes a vehicle requirement. When comparing between viable methods, however, those solutions which cause the least impact to the other subsystems while providing similar communication and operator performance are clearly superior. Standard measurements include data throughput, data storage, and the price, mass, power, and sampling requirements of the sensors.

Similarly hidden in the background of the Summary Problem is the use of the ground equipment such as computers to store and further analyze the Summary data.

Like vehicle impact, this issue is somewhat clouded by the fact that Summary is required, and thus the equipment is required. Often, the ground equipment is fixed, pre-existing infrastructure and so the issue is not what equipment to choose, but what can be done with what is available. The main measurement for this parameter will be the cost involved with creating and maintaining the equipment.

Using the Metrics

Given the specific goals for each mission, these individual performance metrics are weighted to reflect the most important issues. For example, a specific deep-space missions may emphasize the cost of the communication link because of the effort involved to receive weak signals. But a global communications network will be very concerned about the response time since the loss of payload functionality means a loss of business. These metrics can be ranked in priority, or combined into a weighted sum, to provide one general measurement of system performance.

Of course, the driving factor in developing the beacon-based health monitoring problem – which in turn drives the Summary Problem – is cost. Summary solutions which maintain the level of cost savings promised by beacon monitoring will be given overwhelmingly priority. Therefore, in general, the two most important metrics are operator cost and communications cost.

Since this paper is intended to investigate the major issues of the Summary Problem, it is impossible to rank the metrics with any greater precision. Instead, the solutions that are proposed will be measured up against each individual metric. Furthermore, true quantitative measurements have not yet been established, and thus the solutions in this paper must be qualitatively examined.

5. SOLUTION ELEMENTS

Having made some initial attempts to create candidate solutions, and having established the major metrics to evaluate performance, it is now possible to develop the "solution space" of the problem. As defined, there are two main components of a Summary solution: the data that is generated, and the manner in which it is sent to the ground. These categories are further subdivided into the main characteristics that can be adjusted to solve the problem. The major, trade-level elements of the Summary Problem are listed in Table 4.

Transmission Characteristics

The main questions about the data transmission are "How much data?" and "How often is it sent?" Each of these questions are directly reflected in the metrics. As explained below, the "How much?" question is a function of the Summary techniques used by the spacecraft. It can be effectively ignored for the purposes of developing transmission schemes, since the impact to the communications subsystem will be small compared to that required for "normal" telemetry downloads.

The "How often?" question, then, is the fundamental issue in the transmission aspect. In development of the Summary Problem, it was assumed that very infrequent – on the order of weeks – summaries was the best plan. This may not be true; regular, very short, daily updates may in fact provide better performance. Whatever the result, the

Table 4 Engineering Data Summary Solution Elements

Component	**Elements and Sub-Elements**		**Description**	**Examples**	**Costs**
Data Transmission	**No-Contact Duration**		How long to wait between downloads	Never Weekly Monthly On-Demand	On-Board Storage
Data Generation	**On-Board Model**		How the spacecraft estimates its behavior	None Yellow/Red Limits State-Based Limits Transfer Functions	Throughput Reasoning Capability
	Sensor Selection		What information the system provides about itself	None Standard Sensors Commands	Sensor Requirements Data Sampling
	Summary Technique	**Sample Rate**	How often to poll each sensor	Every Minute Event-Based	On-Board Storage Data Transmit Rate Throughput Reasoning Capability
		Comparison	How to relate model and sensor outputs	Alarm Thresholds Error Bounds	
		Manipulation	Number-crunching the data	Statistics	
		Abstraction	Reasoning about the data	Abstracted States	
		Data Selection	What data gets sent to the ground	Nothing Raw Telemetry Abstracted States	

parameter to be traded is the amount of time between data downloads. This parameter is especially important because it directly contributes to the communication cost metric.

As explained in assumption (6), other aspects of the data transmission are not expected to significantly contribute to the performance of a Summary solution and are ignored for the purposes of this study. Since the metrics favor vast reductions in data size, not just transmission speed, such factors such as transmitter frequency, data rates, and similar characteristics are not considered.

Summary Generation Characteristics

The engineering data summary is the product of three elements. Only a few elements may be part of a particular solution, but every solution will typically have all to some degree. First, a set of sensors provides input to the system. These inputs are used in an on-board model to predict expected behavior. The sensor data and model predictions are then compared using a summary technique, which may perform additional processing. The output of the summary technique is the data to be downloaded.

Sensor Selection - This element describes the information available to the Summary system. In general, this refers to the numbers, types, and locations of the sensors used by the vehicle. Often, the spacecraft sensors are determined and positioned according to the requirements of other spacecraft subsystems, and thus the Summary method may not the ability to make trades. However, it is confidently expected that the ability to choose and place telemetry sensors will greatly enhance a Summary's performance, and thus sensor selection will become a necessary element of effective Summary solutions.

In addition to "normal" sensors, such as temperature, voltage, and current, it may prove helpful to include other information about the vehicle, such as the inputs from cameras, the available CPU memory, and state information such as what components are active at a given time.

The decisions made in sensor selection directly impact the spacecraft design in terms of adding hardware; each sensor has mass, volume, power, thermal and data handling requirements associated with it. Adding non-traditional sensors may also require added computational capabilities.

On-Board Model - This element describes the ability of the vehicle to determine the detailed state of each of its components. From a Summary standpoint, this parameter reflects the level of on-board reasoning and modeling. For example, the standard practice of defining "abnormal" and "emergency" alarm thresholds for sensor outputs is one form of modeling; the output is abstracted into five ranges (dangerously low, abnormally low, normal, abnormally high, and dangerously high), where the boundaries are based on an understanding of the expected output. A slightly more sophisticated on-board model would be a set of adjustable limits, altered for different modes - such as sunlight and eclipse - and changed over time to reflect natural environmental degradation. A much more sophisticated model would consist of a set of transfer functions for each component that predicted outputs, given various inputs and the current state of the vehicle.

It must be pointed out that "sophisticated" does not imply "better". Increasing the quality of one parameter does not necessarily indicate a more optimal Summary solution. Increasing the complexity of the on-board model impacts the rest of the spacecraft, which may affect the performance or feasibility of other designs. For example, detailed on-board models require highly capable computers to handle the throughput and reasoning requirements. On-board models also require varying degrees of data storage. The issue of "better" is resolved using the metrics.

Summary Technique - The actual data manipulation is called the Summary technique; the information provided by sensors and the model are compared and transformed in order to create the Summary. There are several key parameters to the Summary technique, all of which are highly coupled. Their functions are distinct enough to be described separately, even if altering one sub-element causes profound changes in the others.

One of the basic issues to be addressed is the *sample rate* of the sensed data. This decision, which may differ for each sensor, indicates the amount of information available to the Summary. It also affects the ways in which data is used to create the Summary.

Another question is what to do with the information generated by the on-board model. *Comparisons* should be made between the model and the sensor outputs, but what sorts? The answer relies greatly on the nature of the model; abnormal and emergency alarm thresholds are designed for specific comparisons, but for a transfer function model, it must be determined to what degree the model output and the sensor output should match.

Also, given the model outputs and the sensor outputs, there are many data *manipulations* that can be performed. Statistics such as extrema, averages, and frequency spectra can indicate whether a component is behaving as expected. A simple curve fit could be derived to compress a time-history into a few parameters. This area is one of the most-explored in engineering data summarization, as the tools are readily available and much of the work done for anomaly isolation applies.

In addition, there may be *abstractions* to be made about this data. Perhaps the performance of a component or a subsystem would be summarized in a single parameter.

Finally, once all this information has been generated, it is necessary for a *data selection* process to determine which of the information is saved. All or part of the data created by the other processes is put into long-term spacecraft memory for eventual download.

In all, Summary techniques contribute to the cost of operations in several ways, most notably the amount of data that must be generated. In addition, design of these elements levy requirements on the spacecraft processing capabilities and reasoning abilities.

Shortcomings to This Method

The primary shortcoming in this approach for defining the elements of the Summary problem is that the boundaries between the elements are not clearly distinguishable. For example, the existence of abnormal and emergency alarm thresholds is considered part of the on-board model, but use of the thresholds is part of the comparison sub-element of the Summary technique. This distinction is quite arbitrary. This problem is not simply a matter of choosing better categories; the complex nature of the Summary problem defies simple classification. However, these categories do provide an effective means of creating more workable sub-problems, an import first step in methodically addressing the Summary Problem.

6. Conclusions & Future Work

Beacon-based health monitoring extends the capabilities of spacecraft mission operations and promises to significantly reduce the cost of operations. However, in order to fully realize these costs, additional work is needed to ensure that in the long term, operators retain understanding about the vehicles. Engineering data summarization is a necessary part of beacon monitoring, giving operators the context they need about the specific components of each spacecraft.

Work has begun on a methodology for creating effective Summary approaches. Quality, timeliness, and cost measures have been established, with particular emphasis on the cost of using human operators and communications resources. The necessary elements of a Summary solution have been identified and further subdivided into their composite parts.

As shown in Table 5, five "strawman" solutions have been proposed, classified according to their solution space elements, and related according to their metrics. Actual evaluation of these solutions would depend on the specific qualities of the mission in question.

As mentioned above, this paper is the start of an ongoing study of engineering data summarization. The next step is to better quantify the metrics involved, and to apply these metrics to the initial candidates. Once the relationships are established between solution parameters and the various metrics, true trades can be performed to identify which elements of the Summary solution space are most vital to effective solutions. It will be insightful to use information from existing and proposed programs to generate real metrics.

Another look is warranted into how the solution space has been partitioned, since the coupling and nebulous distinctions between different elements impedes the ability to perform trade studies. Perhaps a few more candidates could help identify the key, independent elements.

Given those developments, true trade studies could be performed on candidate solutions for engineering data summarization. Test cases must be created to provide realistic examination of various Summary approaches. Stanford's SAPPHIRE microsatellite and its "Al Wood" engineering prototype are candidates for true operational studies.

More work is also needed on the initial problem statement. Are these all the tasks operators perform? Is the ability to distinguish quirks from faults the only real requirement of a Summary? These answers will be found through interviews with spacecraft operators from a number of organizations.

Another assumption worth challenging is the performance of new communications technologies. Perhaps something like a laser communications link, used sparingly, could restore much of the telemetry archive, thereby alleviating much of the Summary Problem.

Table 5 Comparison of Initial Summary Solutions Using Qualitative Cost Metrics

Solution	**Significant Elements**	**Favorable Metrics**	**Unfavorable Metrics**
Reduced Sample Rate	No-Contact Duration Sensor Selection Sample Rate Data Selection	On-Board Storage On-Board Processing Sensor Requirements	Operator Man-Hours Length of Contact Data Downloaded
On-Board Database	No-Contact Duration	Time Between Contacts Data Downloaded	Operator Man-Hours Length of Contact On-Board Storage
Statistical Summary	Manipulation	Length of Contact Data Downloaded	Operator Man-Hours
Alarm Threshold Summary	On-Board Model Comparison	On-Board Storage Time Between Contacts Data Downloaded Length of Contact	Operator Man-Hours
Model-Following Summary	On-Board Model Comparison Selection	Operator Man-Hours Length of Contact Data Downloaded	On-Board Processing Cost of Equipment

Finally, this paper assumes that most of the Summary is performed on board the spacecraft. But is there any benefit to a second Summary performed on the ground? Perhaps some of the long-term trends and model corrections could be automatically performed by ground systems using the Summaries downlinked over several months.

ACKNOWLEDGEMENTS

The author would like to thank his friends and colleagues in the Space Systems Development Laboratory, especially the past and current members of the SAPPHIRE development team. Special thanks goes to Christopher Kitts for his tireless devotion to making clear sense out of these incoherent ideas, and to Professor Robert Twiggs for his continued efforts to keep this laboratory moving. This work was performed in partial satisfaction of graduate studies at Stanford University.

REFERENCES

[1] Mike Farmer and Randy Culver, "The Challenges of Low-Cost, Automated Satellite Operations," *Proceedings of the 31st International Telemetering Conference*, Las Vegas, NV, October 30-November 2, 1995

[2] E. Jay Wyatt and John B. Carraway, "Beacon Monitoring Approach to Operations," *Reducing the Cost of Spacecraft Ground Systems and Operations* , Oxfordshire, U.K., September 27-29, 1995.

[3] Schultz, Michael, "Lifeline: A Concept for Automated Satellite Supervision," *Proceedings of the Software Technology for Satellite Autonomy Workshop*, June 22-25, 1993.

[4] Richard A. Lu, Tanya A. Olsen, and Michael A. Swartwout, "Building 'Smaller, Cheaper, Faster' Satellites Within the Constraints of an Academic Environment," *Proceedings of the 9th Annual AIAA/USU Conference on Small Satellites*, September 19-22, 1995.

[5] Michael A. Swartwout and Christopher A. Kitts "A Beacon Monitoring System for Automated Fault Management Operations," *Proceedings of the Tenth Annual AIAA/USU Small Satellite Conference*, September 16-19, 1996.

[6] Michael A. Swartwout and Christopher A. Kitts, "Automated Health Operations for the SAPPHIRE Spacecraft," *ITC/USA '97: Proceedings of 33rd Annual International Telemetering Conference*, October 30-November 1, 1997.

[7] Richard J. Doyle, "Determining the Loci of Anomalies Using Minimal Causal Models," *International Joint Conference on Artificial Intelligence*, August 1995.

[8] Christopher A. Kitts, "The ASSET Client Interface: Balancing High Level Specification with Low Level Control," *Proceedings of the 1998 IEEE Aerospace Conference*, March 21-28, 1998.

[9] John B. Carraway and Gael F. Squibb, "Autonomy Metrics," *Fourth International Symposium on Space Mission Operations and Ground Data Systems*, September 16-20, 1996.

__Michael Swartwout__ is a Doctoral Candidate in Aeronautics & Astronautics at Stanford University. His research is in the area of engineering data summary. He is involved in SSDL's Automated Space Systems Experimental Testbed (ASSET) operations research system, developing the system architecture and managing the blackboard tool that forms the basis of ASSET. He is also the Project Leader for SAPPHIRE, SSDL's first student-built spacecraft. On SAPPHIRE, he is co-investigator of the Beacon-Based Health Monitoring Experiment. He earned a BS in Aeronautical & Astronautical Engineering from the University of Illinois at Urbana-Champaign in 1991, and a MS in the same, from the same, in 1992.

EXPERIMENTS IN AUTOMATED HEALTH ASSESSMENT AND NOTIFICATION FOR THE SAPPHIRE MICROSATELLITE

Michael A. Swartwout[*], *Carlos G. Niederstrasser*[†], *Christopher A. Kitts*[‡],
Rajesh K. Batra[*] *and Kenneth P. Koller*
Stanford Space Systems Development Laboratory
496 Lomita Mall
Stanford, CA 94305-4035
Fax: 1 (650) 725-3377
rcktboy@leland.stanford.edu

ABSTRACT

As part of its space operations research program, Stanford University's Space Systems Development Laboratory (SSDL) is implementing an automated state of health assessment and notification system for spacecraft. On board the spacecraft, this system consists of software that filters telemetry to derive a health assessment and a periodic beacon that broadcasts this assessment to the Earth. Throughout the world, a network of low cost receiving stations receive the beacon signal and relay it to a central mission control center via the Internet. At the mission control center, a suite of software responds according to the value of the health assessment; appropriate responses may include operator notification, automatic groundstation rescheduling to accommodate new health operations, and intelligent retrieval of appropriate operational documentation.

Conceptually, this system acts as an automated mapping from spacecraft state to high-level operator response. It is being developed as a means to reduce the cost of space missions by drastically reducing the operator hours and communication bandwidth committed to nominal health monitoring. Validation of this system is being performed experimentally on the Stanford AudioPhonic PHotographic InfraRed Experiment (SAPPHIRE) microsatellite as operated through a real-world space operations system under development at Stanford University. This system includes a number of university-built microsatellites and groundstations. System level performance metrics will include cost of nominal monitoring, timeliness of anomaly notification, and quality of the health assessment.

This paper reviews the overall design of the health assessment system, detailing the unique aspects of the SAPPHIRE flight software and the Beacon Automated Contingency-in-Orbit Notification (BACON) receiving station. Results of initial, pre-launch system testing are also presented.

INTRODUCTION

Declining federal outlays for space projects and increased market pressures on commercial ventures are forcing space missions to find ways to reduce cost. Since mission operations can consume a significant portion of the overall budget – especially for long-term programs – special attention is being paid to lowering these costs. A commonly-held assumption in the space industry is that automation can lead to significant operational cost

[*] Doctoral Candidate, Aeronautics & Astronautics, Stanford University
[†] Engineer's Candidate, Aeronautics & Astronautics, Stanford University.
[‡] Doctoral Candidate, Mechanical Engineering, Stanford University

reductions without drastic cuts in mission performance. The Space Systems Development Laboratory at Stanford University is performing an experiment in automated health monitoring to assess these claims.

The core of this experiment is a health-indicating beacon, which essentially maps vehicle state to high-level operator response. The key elements of this "beacon monitoring" approach are: on-board health assessment, low-bandwidth signal transmission, low-cost automated receiving stations, and an "operator on call" response system. The beacon monitoring method aims to reduce cost in two ways: operator workload is reduced by performing routine health monitoring using automated systems; and communication requirements are reduced by migrating this automation to the spacecraft.

After reviewing the laboratory's satellite and operations architecture, this paper outlines the beacon monitoring approach undertaken by SSDL. Sections are devoted to flight software implementation and the design and development of the BACON automated receiving station. The final sections of the paper describe preliminary operational testing using the SAPPHIRE microsatellite and plans for future study.

THE SPACE SYSTEM DEVELOPMENT LABORATORY RESEARCH PROGRAM

SSDL was chartered in 1994 to provide world class education and research in all aspects of spacecraft design, technology, and operation. To achieve this goal, SSDL members enroll in a comprehensive academic program composed of coursework, project experience and research investigations. As one of their investigations, SSDL is actively involved in research in spacecraft operations and automation.

The Satellite Quick Research Testbed (SQUIRT) Microsatellite Program - The SSDL SQUIRT program [1] is a yearly project through which students design and fabricate a real spacecraft capable of servicing low mass, low power, state-of-the-art research payloads. By limiting the design scope of these satellites, the project is simple and short enough so that students can see a full project life cycle and are able to technically understand the entire system. Typical design guidelines for these projects include using a highly modular bus weighing approximately 25 pounds, a hexagonal form that is roughly 9 inches high by 16 inches in diameter, amateur radio communications frequencies, and commercial off-the-shelf components. Missions are limited to about one year of on-orbit operation. Since little money is available for operations, a highly automated mission control architecture is being developed.

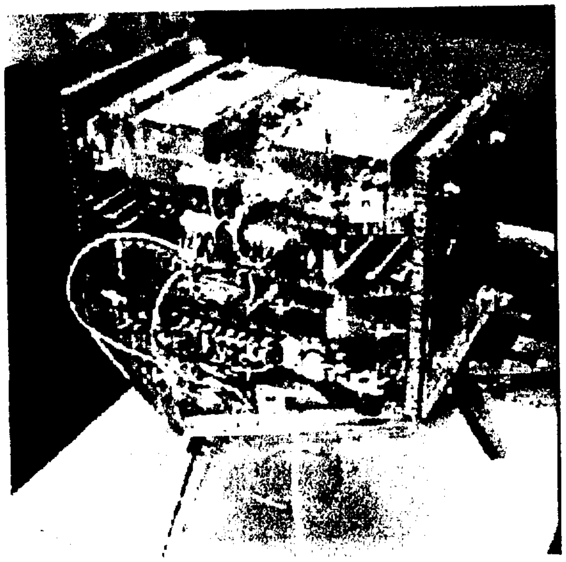


Figure 1 - SAPPHIRE during assembly

The Stanford Audiophonic Photographic Infrared Experiment (SAPPHIRE) Microsatellite - Shown in Figure 1, SAPPHIRE is the first SQUIRT spacecraft [2]. Its primary mission is to characterize the on-orbit performance of a new generation of infrared horizon detectors, in addition to flying two student instruments, a digital camera and a voice synthesizer. Student research interests are also driving experiments in nontraditional sensing and automated operations. SAPPHIRE is hexagonal, measuring 17" across its longest dimension and 13" high. It is primarily constructed of commercially available equipment: the communications subsystem consists of terrestrial amateur radio kits and terminal node controller enabling AX.25 communication in the

2m and 70cm bands; the vehicle CPU is a Motorola 68000 series microcontroller with 256k ROM and 1024k RAM; the aforementioned student payloads are modified off-the-shelf products. In addition, certain mission-critical elements are composed of or modified with some space-qualified elements, such as the space-rated 10-cell NiCad battery pack and radiation-hardened memory for the CPU. SAPPHIRE is being completed by a core of volunteers and research students, and is currently undergoing final preparations for launch. It will be launched as a secondary payload. Although it has not yet been launched, both SAPPHIRE and its fully-functional engineering prototype are available for operational testing and experimentation.

The Automated Space System Experimental Testbed (ASSET) System - The ASSET system [3] is a global space operations network under development within SSDL. The first goal of this system is to enable low-cost and highly accessible mission operations for SQUIRT microsatellites as well as other university and amateur spacecraft. The second goal of this system is to serve as a comprehensive, low inertia, flexible, real-world validation testbed for new automated operations technologies. Figure 2 shows a high level view of the ASSET mission architecture. The basic components include the user interface, a control center, ground stations, communications links, and the target spacecraft. During the current developmental phase, a highly centralized operations strategy is being pursued with nearly all mission management decision making executed in the control center. These tasks include experimental specification, resource allocation throughout the ground and space segment, anomaly management, contact planning, data formatting and distribution, and executive control.

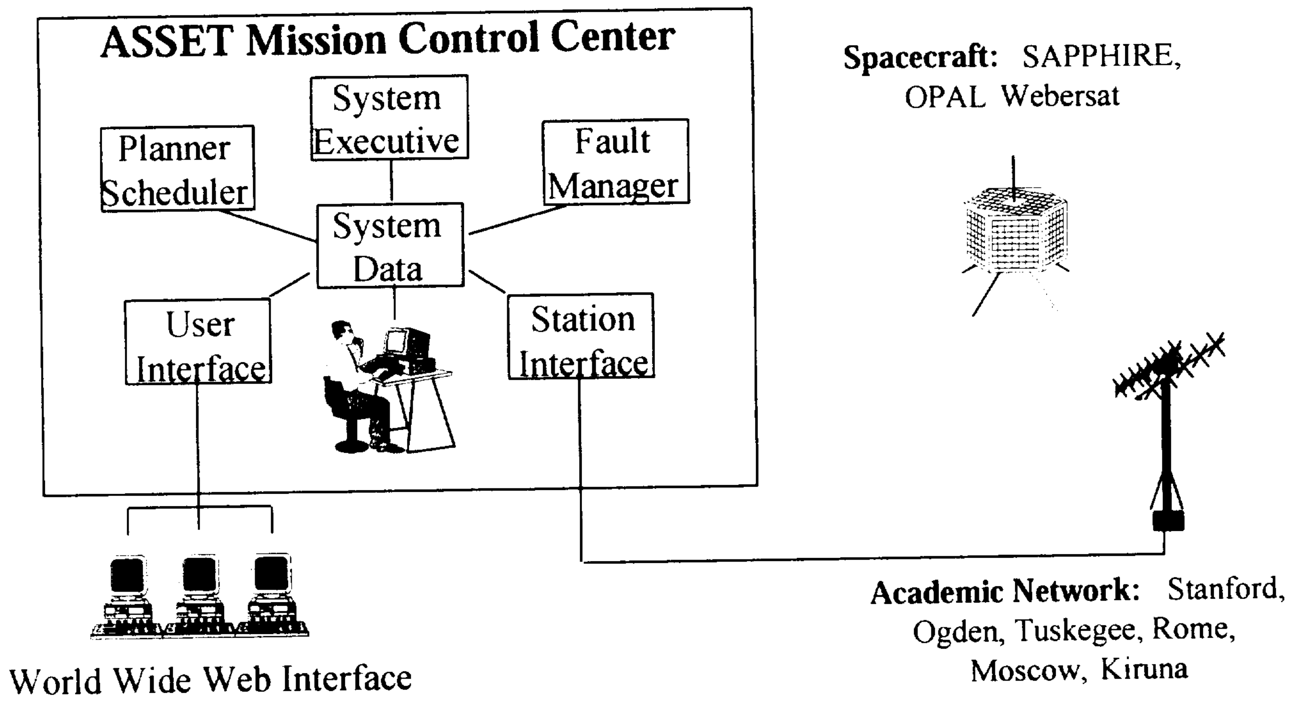

ASSET Mission Control Center
System Executive
Planner Scheduler
Fault Manager
System Data
User Interface
Station Interface
Spacecraft: SAPPHIRE, OPAL Webersat
Academic Network: Stanford, Ogden, Tuskegee, Rome, Moscow, Kiruna
World Wide Web Interface

***Figure 2** - The ASSET Space System Architecture*

SAPPHIRE and all future SQUIRT satellites will be operated through ASSET. In addition, controllers for a number of other university and amateur satellites have expressed in becoming part of the system. As for ground stations, the Ogden and Stanford ground stations are the first two facilities to be included. Several other stations throughout North America and Europe have been identified for future integration.

BEACON-BASED HEALTH MANAGEMENT DESCRIPTION

A beacon-based health management concept was first presented in a U.S. Air Force study, Lifeline [4]. It is currently a flight experiment aboard NASA's Deep Space 1 [5] and is one of the key technologies for future NASA deep space missions [6]. This concept is being prototyped as a part of the SAPPHIRE mission [7]; its main features are summarized, below, and the new elements are further detailed. The signal flow for the SAPPHIRE implementation is presented in Figure 3.

SAPPHIRE Health Monitoring - SAPPHIRE will monitor its own sensors, comparing measured values with expected values in a state-dependent limit table. Certain measurands will be validated by aggregate analysis. For example, the vehicle's configuration prevents all solar panels from seeing the Sun at once; if solar panel measurements indicate that all panels are generating current, then there is good reason to believe that the current sensors are malfunctioning. These modest steps provide SAPPHIRE with an anomaly detection system far more mature than most spacecraft. Software implementation is described in a later section of this paper.

Depending on the seriousness of the limit violation, the spacecraft state is assessed to be one of four values. For example, when measurands are within limits, the spacecraft is judged to be *Normal*. Out-of-limits with moderate impact, such as an overheating camera, is considered an *Alert*. Out-of-limits that can rapidly jeopardize the mission elevates the health status to *Critical*. Finally, *Emergency* condition is defined to be an unexplained computer reset. Note that the rules by which measurands trigger the modes, and the limits for each, are defined by the operations team. This ensures that beacon modes are a mapping from spacecraft state to operator action.

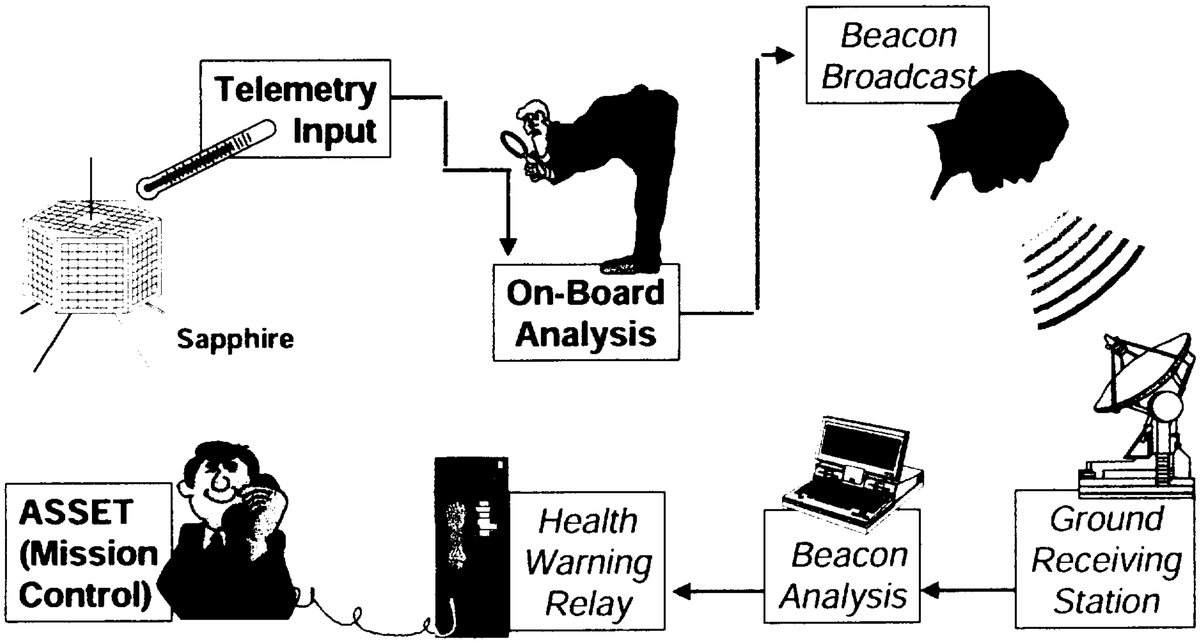


Figure 3 *- SAPPHIRE Health Beacon Signal Flow*

Health Beacon Transmission - The beacon is a pulse-modulation of the main transmitter carrier, with different pulse widths defined for a one bit and a zero bit. The total transmission time of the beacon message is less than one second. The message is broadcast whenever beacon operations are active, nominally at one minute intervals. Therefore, spacecraft health is continuously monitored and the health indication is available anytime the spacecraft is within range of a receiving station.

Receiving Station – SSDL has developed the prototype BACON receiving station, more fully described below. Based on a schedule provided by ASSET, it listens for SAPPHIRE transmissions. If a beacon signal is received, the pulse modulation is converted into bits and this information is time-tagged and sent via electronic mail to the ASSET mission control.

Mission Control Center – Once mission control receives a beacon monitoring update from a remote station, it logs this information and then takes appropriate action. Depending on the health assessment, there are varied responses, from storing the update in the system database to paging the operator on call and rescheduling the network to contact and recover a failed satellite.

IMPLEMENTATION - SAPPHIRE SOFTWARE

Modifications in flight software have given SAPPHIRE the ability to monitor itself, using a conventional limit-checking approach for health assessment. The table/beacon software is embedded in Sapphire's operating system. Called Chatterbox [8], it is a student-developed bulletin board system with a UNIX-like interface. This C-based platform is divided into hardware drivers and user interface modules. The modifications necessary to implement on-board health monitoring have been primarily in the creation of new data constructs, specifically limit-checking tables and virtual sensors.

Once every sample period (a commandable value, typically ten seconds), the operating system compares each of the designated sensors against high and low limits stored in a table; if the value exceeds the limits, then the system is instructed to perform a series of commands. These commands are built-in to the table and can be tailored to specific events. For example, an indication that a payload is too cold would result in a command to turn it on in order to generate heat. Such an event would also change the beacon message to *Alert*. If the battery voltage drops too low, however, it would trigger a sequence of commands to put the vehicle into a safe mode and send the *Critical* beacon message. Example table entries are shown in Table 1.

Channel	Low	High	Command	*Translation*
5	12.0 V	16.0 V	Jump to Table 8	*If battery voltage (channel 5) exceeds range, activate Safe Mode (table 8)*
1	-	50 mA	sensor set 32 step 1	*If solar panel 1 current (channel 1) is above threshold, increment the counter panels_in_sunlight (virtual channel 32)*
24	278 K	-	os pins set 2 1	*Turn on camera (pin 2) if the temperature (channel 24) is below 278 degrees Kelvin*
24	278 K	318 K	os beacon message 01	*Set beacon to "alert" if camera is out of temperature range*

***Table 1**: Sample SAPPHIRE Table Entries with Explanations*

As designed, the table system is extremely flexible; the software contains a series of pre-built tables to monitor major vehicle states, but these entries can be changed. The channels to examine, the low and high limits, and the triggered responses are all fully adjustable by ground command. Alarm thresholds can therefore be fine-tuned on-orbit to account for environmental degradation and other mid-mission changes.

Another important contribution of this flexible flight code is the ability to perform mode-based limit-checking. For example, there are different expectations for component temperatures and battery performance during sunlight and eclipse; the ability to tailor the alarm thresholds to specific modes allows for more accurate health monitoring. SAPPHIRE accommodates mode-dependent limit checking by enabling the operating system to add and delete table entries "on the fly" and by creating "virtual channels" containing abstracted information about the spacecraft. Such abstractions include the number of solar panels in sunlight and whether or not the vehicle is in eclipse. Again, these virtual channels can be customized after launch, based on the situations encountered during flight operations.

Another issue which the software must address is how to identify spurious state transitions. It is undesirable for a sensor value oscillating near an alarm threshold to trigger repeated transitions from one state to another. Such transitions may be the result of nothing more than sensor noise. To mitigate this problem, the SAPPHIRE code implements persistence counters. A sensor value must persist beyond the threshold for a certain number of cycles before it is considered to be out of limits; similarly, it must persist within the threshold boundaries to be normal again. Like the other elements of the table system, the persistence counter can be set for each entry. For clarity, the persistence values were not shown in Table 1.

IMPLEMENTATION - BEACON RECEIVING STATION

If a beacon receiving station is to be cost-effective, it was determined that total implementation costs had to be approximately an order of magnitude below a "standard" ground station. For communicating with SAPPHIRE, a standard station includes hardware for two-way communications (Mode-J), a TNC for packet radio, and one or two computers for tracking and data. Total cost for this setup is approximately $10,000.

These cost constraints impose severe limits on the kind of hardware that can be used for the station. Additional constrains are imposed by a need for automation and remote location. Instead of a fully tracking antenna, a much lower gain omni-directional antenna is used. The antenna connects to a computer-tunable commercial receiver able to detect an RF carrier and convert it to an audio signal. The audio signal is then fed to a standard sound card on a Pentium-based computer. The computer must be connected to the internet, so that the station can send and receive data from the ASSET mission control center.

At the heart of the station is a Windows95 program that monitors incoming data from the receiver. As the audio signals come in through the sound card, they are transformed to a frequency spectrum representation by use of a Fast Fourier Transform algorithm. Since the data rate (5 Hz) is much lower than the audio signal frequency (3 kHz) the signal comes across cleanly without any modulation. After accounting for the satellite's transmit frequency and any Doppler shift, the software must simply look at the appropriate audio frequency, and determine whether enough power is present to represent a true signal.

Once the software knows that a signal is present, it is examined over time to determine the on-off pattern present. This pattern is then directly translated to a beacon code that has previously been defined in the information for the relevant satellite. The code, the time of receipt, the station ID, and the satellite ID are immediately e-mailed to mission control to be processed appropriately.

To monitor the skies effectively, the beacon receiving station depends on ASSET to provide it with a visibility schedule of the satellites in the system. It is as important for ASSET to know when a satellite was *not* heard, as it is to be told what code a satellite is broadcasting. The beacon station will also notify the system if the satellite was heard, but no sense could be made of the message. This could be due, for instance, to packet communications taking place between the satellite and the ground. Finally the station will treat itself as a "satellite" and send regular beacon messages to ASSET informing the system of its own state of health, and whether human interaction at the station is needed.

A prototype of the BACON station has been developed at SSDL. It has demonstrated capabilities to detect beacon signals from SAPPHIRE, convert beacon transmissions into information bits, and to forward this time-tagged information to ASSET via electronic mail. The BACON station is available for use in the preliminary operations testing, described in the following section.

PRELIMINARY TESTING

The beacon monitoring system of SAPPHIRE has been ground-tested. Not only have these tests demonstrated the ability of the system to function as expected, but useful data has been gathered concerning the cost savings from beacon operations. SAPPHIRE or its engineering prototype can be stacked in flight configuration and operated remotely at Stanford, using either the main SSDL ground station or the portable testing unit. Flight-like conditions can be simulated by restricting communications access to the vehicle to windows of opportunity reflective of a low-Earth orbit.

For these tests, two independent teams of SAPPHIRE operators were assembled. One team performed nominal operations, as if there were no health beacon and contacting the vehicle once per day. The second team relied on the BACON station to monitor the health beacon, contacting the vehicle only once per week. The test was conducted for one week of operations, during which time a third party created a "fault" by power cycling the vehicle. The purpose of this test was to measure the operator effort required to perform nominal health monitoring and the communications resources needed to identify the presence of an anomaly.

During this test, preliminary results indicate that the beacon-based health monitoring approach offers significant savings over the operator-intensive method. Not only did the automated approach indicate the presence of a "fault" using fewer man-hours and communication time, but the automated operations team was aware of the problem's existence hours before the nominal operations team!

CONCLUSIONS AND FUTURE WORK

The use of a low-power beacon and low-cost automated listening stations is a viable solution to the new challenges in spacecraft health management. Automation of limit detection and migration of this process to the spacecraft will significantly reduce the spacecraft-to-ground communication link and the time spent by operators on routine health management tasks. This beacon-based health management approach has been initially demonstrated in a ground test of the SAPPHIRE spacecraft using the ASSET operations architecture.

The preliminary tests described in this paper will be expanded to cover longer periods of operations and additional operating conditions. More "unknown" faults will be injected into the system; doubtlessly additional "true" anomalies will be discovered over the course of operations. Also, longer operational periods will help explore the potential long-term effects of automated monitoring, such as the absence of an exhaustive ground telemetry archive and the relative unfamiliarity operators have with vehicle nuances. Such effects are the subject of additional research within SSDL. Additionally, SSDL's second satellite, OPAL, will be fitted with appropriate beacon software. This will allow testing of multi-satellite scenarios.

Ultimately, this health monitoring approach will be tested on-orbit during SAPPHIRE's flight. Until then, enhancements will be made to the ASSET system and to the BACON station, especially to accommodate Doppler shift and loss of signals. Other than some minor changes to the signal parameters, the SAPPHIRE software is frozen and ready for all future tests.

ACKNOWLEDGEMENTS

The authors wish to thank of the SAPPHIRE team for their dedication and efforts. The Jet Propulsion Laboratory and NASA Ames Research Center are acknowledged for their support and assistance with this research project. Finally, appreciation is extended to all of the institutions and organizations that have participated in, contributed to, and provided feedback concerning various aspects of the ASSET system. This work has been performed in partial satisfaction of graduate studies at Stanford University.

REFERENCES

[1] Kitts, Christopher A., and Robert J. Twiggs, "The Satellite Quick Research Testbed (SQUIRT) Program," Proceedings of the 8th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 16-22, 1994.

[2] Lu, Richard A., Tanya A. Olsen, and Michael A. Swartwout, "Building 'Smaller, Cheaper, Faster' Satellites Within the Constraints of an Academic Environment," Proceedings of the 9th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 19-22, 1995.

[3] Kitts, Christopher A., "A Global Spacecraft Control Network for Spacecraft Autonomy Research." Proceedings of SpaceOps '96: The Fourth International Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany, September 16-20, 1996.

[4] Schultz, Michael, "Lifeline: A Concept for Automated Satellite Supervision," Proceedings of the Software Technology for Satellite Autonomy Workshop, Albuquerque, NM, June 22-25, 1993.

[5] Sherwood, Rob, Jay Wyatt, Alan Schlutsmeyer, and Mike Foster, "Flight Software Implementation of the Beacon Monitor Experiment on the NASA New Millenium Deep Space 1 (DS-1) Mission," Second International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Oxfordshire, United Kingdom, July 1997.

[6] Wyatt, E. Jay, and John B. Carraway, "Beacon Monitoring Approach to Spacecraft Operations", Reducing the Cost of Spacecraft Ground Systems and Operations, Oxfordshire, United Kingdom, September 27-29, 1995.

[7] Swartwout, Michael A., and Christopher A. Kitts, "A Beacon Monitoring System for Automated Fault Management Operations", Proceedings of the Tenth Annual AIAA/USU Conference on Small Satellites, Logan, Utah, September 16-19, 1996.

[8] Batra, Rajesh K., "The Design of a Highly Configurable, Reusable Operating System for Testbed Satellites," Proceedings of the 1997 AIAA/USU Conference on Small Satellites, Logan, Utah, September 15-18, 1997.