9^{th} Annual
AIAA/Utah State University
Conference on Small Satellites
September 18-21, 1995
Logan, Utah

A System Architecture to Advance Small Satellite Mission Operations Autonomy

Sam Siewert
Linden H. McClure

Faculty Advisor:
Elaine Hansen

Colorado Space Grant College
University of Colorado at Boulder
Boulder, Colorado 80309-0520

Abstract

A system architecture concept that supports automation of small satellite mission operations functions is presented along with a working prototype. The concept and prototype provide an automation framework which enables much more advanced levels of operational autonomy than is typical of current systems. This scalable architecture provides a risk mitigating stepwise approach toward achieving high levels of operational autonomy to enable significant operations cost savings through mission performance management by exception. The prototype consists of an October 1997 manifested Space Shuttle Hitchhiker payload along with the software and hardware described in this paper.

Introduction

Future generations of satellites and deep space probes must be considerably more operable and autonomous to enable continuing exploration, science investigation, and commercial utilization of space. The need for greater operability and autonomy stems from shrinking governmental budgets to support scientific planetary exploration missions, commercial interest in deploying large constellations of communications and imaging satellites, and the scientific goals to globally survey the earth. Greatly decreased operations costs will not only enable more scientific missions, but will also enable operations systems to be scalable to meet commercial goals such as operating hundreds of communications satellites in low earth orbit constellations [1].

Traditionally, teleoperated satellites include some automated functions both onboard and in the ground segment of operations; however, the human attention required for performance maintenance, resource management and planning alone still remains fairly high. Furthermore, the data rate associated with most low earth orbit satellites has increased rapidly over the past twenty years such that human monitoring of telemetry is becoming less feasible (due to information overload). At the same time, advances in sensor technology, actuator technology, intelligent interfaces, science instruments, and embedded microprocessors have also made higher levels of automation much more feasible [6]. Unfortunately, many of the functions which could be automated are not automated due to many unique factors of the satellite industry and environment compared to commercial computer operating systems. These factors include relatively infrequent usage, missions with large operational budgets, a risk-adverse culture, lack of competition, small

scale deployment, and novel aspects of the space environment.

Some of the factors which have contributed to a lack of mission operations automation are in fact no longer significant or existent today, yet the level of automation of mission operations has not changed dramatically. A typical mission operations team for a scientific or commercial payload in low earth orbit would include tens of operators fulfilling functions such as telemetry monitoring, planning, scheduling, resource management, performance analysis, anomaly resolution, navigation and maneuver planning. Much of this operator workload is not directly related to mission goals or the payload and is simply an artifact of operational systems which cannot be evolved to incorporate higher levels of automation over the life of a single mission, over a number of missions, within a scalable embedded system, or within a constellation of satellites.

Most mission operations concepts include nearly continuous explicit human monitoring from centralized control centers rather than management by exception by distributed personnel. The ability to manage a satellite by exception within a distributed computing environment where the magnitude of human attention required for maintenance is much closer to ones rather than tens of operators, would significantly reduce life cycle cost of small satellites and enable the deployment of large constellations and frequent scientific missions. The architecture presented here provides an approach toward achieving increased operational autonomy with stepwise advancement which removes much of the risk associated with less flexible designs that target a specific level of autonomy.

More frequent missions will further drive automation, increase commercial profitability, stimulate larger scale production of space hardware and software, and help motivate the development of standardized reusable mission operations systems. The resulting economies of scale can enable a phenomena which greatly advanced terrestrial computing systems. These systems have evolved from expensive, high maintenance mainframe computers toward powerful desktop workstations that only require occasional maintenance. While space systems are more complex than terrestrial systems, to a great extent, the same phenomena could be initiated by the small satellite mission operations system architecture presented in this paper.

Operations Concept

The scalability of mission operations without significant automation is limited. Assuming that in the best case scenario, a single small satellite would require a team of ten operators, not including payload operators, and that the number of operators scales linearly with the number of telemetry points to monitor (a large assumption), then a constellation of hundreds of satellites would require thousands of operators and an inordinate operations budget. Furthermore, such systems are likely to require many more operators due to non-linearly scaling workload associated with interactions and dependencies which can result in complex behavior and even unexpected emergent behavior. We hypothesize that automated methods of focusing attention, abstracting information, detecting events such as faults and opportunities, reacting to detected events, and performing onboard automatic replanning could greatly decrease the workload that is placed upon human operators, even with large volumes of telemetry at high data rates.

A common objection to automation is that automation itself incorporates even more complexity into a system. One significant flaw in this objection is that it ignores the fact that humans add significant complexity and unpredictability to the behavior of an overall system when human and computer interaction is considered. Functions allocated to human operators may in some cases be performed far more optimally than with automation, but the ability to predict performance is nearly impossible. Studies show that human operators are a source of faults, and often faults due to human operators are difficult to trace or resolve [10]. While it is true that automation does add complexity to the system, this is simply a testing and validation problem. Automation of functions has the distinct advantages that it can be validated, and is much more reliably scalable than systems which rely upon extensive human-computer interaction at low levels of abstraction.

A fundamental concept of the architecture presented here is that development of automation leading toward greater operational autonomy is stepwise, with incorporation of a testbed to continuously validate and revalidate the system as it is evolved. Therefore, automation of tasks performed initially by humans who can bring significant common sense reasoning and cognitive abilities to bear upon ill-defined problems, can be incorporated as these tasks are better understood and when human cognitive skills have less payoff. As a given mission operations system is evolved toward greater overall autonomy with proven automation, then the scalability of the overall system is greatly increased since the requirements placed upon the human operator to maintain the performance of each subsystem are reliably decreased.

Features of the mission operations system architecture proposed and explored in this paper which lead directly toward operator load reduction and system scala-

bility include:

1. Mission operations management by exception due to:
   (a) automated resource management,
   (b) self-monitoring capability such that fault detection and reaction, or performance maintenance, is mostly autonomous,
   (c) onboard automatic replanning, and
   (d) command and control which may include event driven self-commanding.
2. A hardware and software architecture that supports continuous evolution of automation over mission life cycles and across missions as well as migration of automation from validation testbeds, to ground systems, and to onboard systems.
3. Ability to quickly incorporate new microprocessors, bus architectures, and peripherals into spacecraft embedded systems.
4. Scalability at the level of distributed ground stations, computing environments and constellations of spacecraft.
5. Scalability of onboard microprocessors to multiprocessor systems to support redundancy and distribution of processor load associated with reasonably independent subsystems and payloads.
6. Incorporation of standardized reusable open systems software for ground and embedded systems with many of the application specific functions incorporated into a more generic mission operations extensible and modular kernel.
7. Availability of distributed software/hardware mixed testbeds where "virtual spacecraft" can be configured and used with the ability to simulate and inject faults for testing throughout mission life cycles.
8. Inclusion of security protocols such that the most appropriate distributed resources and qualified personnel can be securely and cost effectively applied to a mission on a demand basis.

The mission operations management by exception achievable with this architecture can lead to an order of magnitude reduction in human attention required. For example, in the case of telemetry monitoring and fault detection, if a subsystem which includes tens of sensors can be automated to self-monitor and produce a single health and status word or indicator, then an order of magnitude reduction is achieved. With implementation and validation of such localized self-monitoring applied across multiple subsystems with regression testing to identify faults identifiable only at the systemic level as subsystems are combined, the load of monitoring is reliably reduced by an order of magnitude or more. A common problem with automated detection is performance with respect to number of false alarms raised compared to probability of failure to detect an anomaly. This is a matter of tuning and effectiveness of the methods chosen for a specific monitoring task. The ability to use the testbed to support tuning and method selection is therefore required.

Reactions to many detected faults are often benign. For example, if the power draw of an instrument is anomalous, and the instrument is in a calibration mode during a non-critical phase of the mission with respect to science data acquisition, then safing the system by switching off the instrument is a straightforward flight rule that is simply automated. The inclusion of a rule-based system allows for such flight rules to be easily captured with rules that include modal and contextual logic along with detection layer decision boundary logic. For example, the identification of safing flight rules would be a first level of reaction automation based upon the detection layer. The only real danger of such fail-safe automation is that false alarms which cause the system to be spuriously safed lead to unnecessary opportunity loss and reduction in overall performance. This is why tuning and validation of the detection layer is so important. More complex reaction automations include resource management, scheduling, and performance maintenance despite faults and system degradation. By automating the simplest, most detectable and highest frequency event-reaction pairs, the telemetry monitoring and commanding load associated with mission performance maintenance is reduced.

While detection to reaction linking can reduce human operational load by an order of magnitude alone, automatic replanning can further reduce the load. In fact, replanning is really a very highly automated type of reaction. One of the significant leverages of this architecture is that automated planning and replanning can be built upon the automated reaction rules and with constraints (which also may be applied to human interfaces to control human computer interaction faults) which can be used as the kernel of the automatic planning system. The rules tied to the detection layer can be used to trigger replanning which can use constraints with well proven planning and scheduling algorithms to generate

command sequences and perhaps even additional rules and constraints. Onboard planning has the greatest automation leverage and risk, but with the systematic approach of achieving higher levels of automation proposed here and supported by the proposed architecture, the risk is greatly mitigated. With the detection and reaction layers of automation, an order of magnitude reduction in work load may be achieved, and with automatic planning based upon these layers, further orders of magnitude reduction can be achieved leading to very high level management by exception of a mission.

System Description

The next generation small satellite mission operations system includes a hardware bus architecture which, much like today's desktop workstations, would allow multiple processors, multiple peripherals, and payloads conforming to industry standard bus interfaces to be quickly integrated for a given mission. This hardware scalability is key to the software required to support the mission operations concept presented here. Such a bus architecture will enable redundancy and scalability, which will accommodate variations in subsystem and payload processing requirements, and will allow the addition of computing power as needed to enable much more onboard processing than is typical of traditional spacecraft computing systems.

The software system architecture proposed enables distribution of functions within the ground computing environment and between the space and ground segments of the overall computing environment. The major systems software functions within this end-to-end distributed computing environment include:

1. Real-time operating system with an extensible and configurable kernel.

2. Digital signal based behavioral and model deviation based fault detection.

3. Rule-based command and control for linking event detection to automated reactions.

4. Automatic planning system with "anytime" algorithms.

5. Internetworking between space segment and ground segment computers.

6. Transparent software and/or hardware based security protocols.

7. Development environment for simulations and mixed hardware/software testbedding with the ability to simulate faults.

8. Distributed ground and embedded system data management schema.

Hardware System

In order to migrate spacecraft control to the flight system, the flight hardware must be able to support the increased software demands that increasing levels of automation and autonomy require. Providing a flight computer capable of heavy processing loads has previously been impossible because of the reliance on radiation-hardened processors which typically are older, less capable technologies. The high cost and long lead times of space rated electronics have discouraged the incorporation of newer technologies as they become available, contributing to the lagging processing power prevalent in spacecraft computers.

Commercial parts By shifting from the conventional frame of thought and utilizing commercial hardware and software elements which are readily accessible and relatively inexpensive, a design can take advantage of the high throughput and capabilities of current technologies. In addition, by allowing the use of commercial technologies in system design, the system designer can incorporate the use of commercial standards, thus providing a system architecture which is scalable, reusable and easily upgradable. The lower cost associated with commercial grade electronics allows migration of a flight design to newer, increasingly capable technologies as they become available. In addition, the lower cost may also permit the construction of several flight testbed systems, allowing development and testing of ideas to occur in parallel. Of course, when using commercial components, potential problems associated with the space flight environment must be overcome [11, 12, 13].

As with any spacecraft, the hardware must be designed to withstand the harsh conditions associated with launch. This is of particular interest when designing a scalable system using off-the-shelf computer hardware, as most commercial environments are not as physically stressful as a launch vehicle and as a result, the computer boards are not designed with the same stress, vibration, shock, and temperature loads in mind. Special construction techniques should be used during board manufacture to provide the mechanical support necessary to survive the rigorous launch conditions. These techniques should include methods for supporting the board to minimize

vibration and flexing, and specifying connector designs which are tolerant of these launch conditions. Maintaining the operating temperature limits of commercial components may place an added burden on the spacecraft thermal subsystem.

Outgassing of the electronics packaging can be a problem when using commercial parts. If outgassing can damage flight system components such as science instrument optics, the use of commercial components may require the use of hermetically sealed enclosures to protect the vulnerable elements.

The radiation environment to which the electronics will be exposed must be analyzed. Certain missions, particularly those of short duration, may experience low total-dose radiation environments which can be easily mitigated by use of shielding. Single event latchups and upsets may be mitigated by utilizing various hardware and software techniques and by incorporating redundant hardware, the added cost of which will in most cases be much less than the cost of radiation hardened circuits. Of course, as in any flight system design, the added weight, volume, and power requirements of the redundant circuitry must be considered.

Some missions may have specific mission requirements which preclude the use of commercial parts. However, the economies of scale provided by the increased numbers of flights enabled by the proposed reusable architecture will provide increased availability of lower cost, high performance space rated electronics suitable for these missions.

Functionality The hardware must support the various demands of increasing levels of automation by providing basic functionality including:

1. Sufficient processing ability for increased software loads.
2. Interrupt support and hardware timers, necessary for real-time operating system support.
3. Sufficient memory, processing power, and system resources to support mission goals.
4. Adequate communications bandwidth to support the transfer of data and commands between the ground system and flight system.
5. Non-volatile memory for storage of code updates and system configuration parameters.
6. Sufficient I/O capability, including both adequate transfer rate and numbers and types of channels.

In addition, to provide the performance that widely varying mission requirements demand, the hardware may support scalability by incorporating additional features including:

1. Direct memory access (DMA) support to allow high speed transfers between system peripherals and memory.
2. Use of appropriate standards such as VMEbus to allow easy incorporation of additional interfaces and multi-processing capability, which can also allow redundancy to protect against a single processor failure.
3. Support for increasingly compact, inexpensive, and power-thrifty memory elements such as solid state disks and standards such as PCMCIA to allow a system to contain far more data storage capability than was previously possible.
4. Support for reusability and reconfigurability; an example being Field Programmable Gate Arrays (FPGA's) used as I/O interfaces.
5. Network support, so that if required, communication between multiple computers in the same spacecraft or within a constellation may use commercial network protocol standards such as TCP/IP.

By providing the ability to reuse existing hardware designs merely by upgrading to the current technology, less time is spent developing and maintaining mission operations software specific to each hardware configuration, thus maximizing the benefits of software reuse.

Software System

As described, the software system will include eight major components to be integrated within an open distributed computing environment. The major emphasis is on scalability so that common systems software can be used throughout the distributed system of ground and space segment computers with tailorability to special segment requirements.

Real-Time Operating System Layer The RTOS (Real-Time Operating System) includes an extensible and configurable kernel such that common kernel software can be incorporated within embedded systems in the space segment and within the ground segment to the greatest extent possible. Current work on RT-Mach, RTEMS, and many other open operating systems which support real-time scheduling can be used to provide this functionality. These operating systems can be used to easily integrate the other major functional layers of this architecture with a common kernel layer that provides standardized abstractions of the processors and hardware interfaces incorporated in the overall system.

The generality of an RTOS in comparison to an application specific executive does mean that greater overhead is placed upon task management for a given application, but it also means greater flexibility, quicker development, and better scalability and ability to support the automation layers of the overall system such as detection and rule-based reaction. For example, the ability to meet deadlines within a real-time system with rule-based inferencing can be greatly simplified by an RTOS scheduler which provides methods to balance rule-based inferencing with other system processing and critical processing with hard deadlines. The reliability of real-time systems incorporating complicated inferencing and planning algorithms can be greatly increased with proven scheduling algorithms [8, 9].

Digital Sensor Based Event Detection Layer This system integrates advanced technologies for digital sensor based fault and event detection, context-based reasoning, hierarchical monitor performance tuning, intelligent monitor-to-action linking, and human computer interfaces. Such functionality is required to enable other autonomy enabling functions such as fault handling, goal-oriented self commanding, automatic replanning, and resource management. This proposal includes enhancements to the specific technologies and methods for applying the technologies in order to provide proven, reliable monitoring automation. The integrated system supports self monitoring in distributed systems, a range of possible human attention, and provides for migration of monitoring between segments of a scalable distributed system. Finally, the system has been designed to simplify the tuning and application of the self monitoring system to a range of systems and missions.

This digital signal based behavioral and model deviation based event and fault detection is a data processing function that will be distributable between the ground and space segments in this proposed architecture. The most significant problem with the use of such automated detection systems is the "tuning" required to achieve beneficial automation for any given monitoring application. This software layer is tunable in terms of decision boundaries, methods, and input transformation fidelity such that overall system detection performance can be optimized in terms of mission requirements for detection performance (false alarms and probability of missed faults). A direct interface between the fault detection layer and the command and control layer should provide that capability to link reactions to detection. This detection reaction linking is fundamental to automation and overall autonomy. Finally, the detection must be context-sensitive such that false alarms which might be raised otherwise due to commands, sequences, and planned mode changes do not raise false alarms leading to unnecessary system load and possibly detrimental incorrect reactions.

In general, fault and event detection methods include: limit sensing, difference-based reasoning, causal reasoning (changes in relationships), reference model deviation, trending analysis, frequency domain entropy-based reasoning, configuration and sequence discrepancy analysis, and many other logical and statistical modeling methods. Difference-based reasoning methods such as distance measures are to be incorporated which include both causal distances (changes in relationships) and simple distances (changes in behavior) [5, 7]. Traditional statistical modeling methods for trending including methods such as ARMA (Autoregression Moving Average) and methods for modal modeling such as linear mixture models will be incorporated and used as benchmarks. Finally, the applicability of ANN (Artificial Neural Networks) as non-linear mixture deviation models and classifiers of time series transformed input is being considered for incorporation.

One of the problems associated with incorporating a large number of detection methods is how to determine when to apply a particular method or a combination of methods to provide the best detection performance. As previously noted, detection methods available include a variety of approaches that can be applied to the sensor time and/or frequency domain output. A support tool for analyzing detection method applicability to specific failure modes will be incorporated in the system. This tool will provide evaluation of the best single method or combination of methods for a specific monitoring application based upon testbed simulation and injection of software and hardware simulated faults into nominal simulations.

Monitors can be related in a hierarchy such that specific sensor monitors can be grouped by component, subsystem, and system. This approach supports mon-

itor tuning at all levels as well as diagnosis and tracing. The structure and the associated relationships would also provide valuable information that could be used by an embedded expert system for autonomous tracing, or a ground based object-oriented database management schema and/or expert system to support diagnosis and tracing that cannot be handled completely autonomously. Finally, this is also key to highly abstracted monitoring since a hierarchical structure of fault information can be directly associated with this monitor structure.

The occurrence of false alarms and false opportunities will be minimized with context-based reasoning. For example, simple mode changes and associated nominal behavior and relational changes can cause false alarms unless the context of the mode change is incorporated into the monitor. Multiple levels of context-based reasoning can be supported with the proposed system, including simple reasoning local to the monitor based upon related sensor values, global information such as subsystem or system mode information provided by external context, and high level rule-based constraints based upon mission and systems engineering parameters. Where this context reasoning is performed depends upon the time criticality of the monitor and the complexity of the context-based reasoning required to provide appropriate monitor performance with respect to detection and false alarms.

Key performance measures of automated monitoring are the false alarm rate and the missed fault rate. While monitor performance is tied to hardware design parameters such as sampling frequency and sensor placement, given a set of samples, the monitoring method performance can be analyzed and tuned. For example, automated fault monitoring is in fact a classification problem, even when faults themselves are not classified, since the fundamental classes are nominal and anomalous. According to the theory of ROC (Receiver Operator Curves) any given classification scheme will result in a tradeoff between the "hit rate" and the "false alarm rate" for a given input sample set. In the case of fault detection, the hit rate is the percentage of correctly classified anomalous samples and the false alarm rate is number of nominal samples incorrectly classified. The ROC tradeoff is based upon imperfections in the monitoring method, (a perfect monitor would correctly classify all samples and therefore raise no false alarms and have a perfect hit rate). The placement of monitor decision boundaries determines where a monitor is on its ROC, and therefore the current tradeoff between hit rate and false alarms can be tuned for a specific detection method.

This method of decision boundary tuning can be used to optimize monitor performance with an individual or set of combined detection methods that provide the best overall hit rate with the lowest false alarm rate for a given monitor. For example, the optimization tool can determine detection method decision boundary synergism for failure mode sets. The ability to look at individual monitor performance and translate this into system level performance is implemented by making use of the monitor hierarchy such that individual monitor, subsystem, and system level monitoring performance can be evaluated and tuned. The proposed monitor performance tuning methods and tools include:

1. A decision boundary tuning method for controlling monitor performance such that the tradeoff between detection sensitivity and the number of false alarms raised for each individual monitor can be optimized.

2. A hierarchical tuning method that will determine the overall system expected alarm rate, false alarm rate, and possibility of failure to detect a true fault based upon performance of lower levels.

3. A tuning tool which will automate the bottom-up and top-down tuning approaches of items 1 and 2 above so that the self monitoring system can be optimized to meet overall monitoring goals.

4. Application of testbed and software simulations, which provide controlled fault injection, to automated fault detection system tuning (items 1, 2, and 3 above).

5. Incorporation of frequency domain automated binning features.

6. Incorporation of confusion matrix analysis and other tuning features.

The self monitoring system does not preclude human monitoring, but does minimize it by elevating the monitoring to a high level. The height of the level can be specified according to mission objectives and the ability to automate monitoring and to link detection with intelligent reactions. At the highest level, the health of the overall system would simply be related to a human operator by a green, yellow, or red indication of mission status. The automated monitoring focuses operator attention only on problems which cannot be handled autonomously or those where operator concurrence is desired (e.g. a red condition). This approach will be integrated with the hierarchy of monitors proposed in the self monitoring system.

Rule-based Command and Control Layer The command and control system will be a layer of systems software above the RTOS kernel that supports the specification of mission operations rules, constraints, sequences and high-level commands. This command and control layer should include an API (Application Programmer's Interface) such that it can be interfaced with sophisticated user interface, interprocess communication, data management, and automation software in the ground segment as well as with reusable device drivers, data management, fault detection, and communications software to be incorporated in the space segment. The rule-based command and control is key for providing a framework for evolving automation, and thus overall operational autonomy, toward higher and higher levels. Simple rules for safing in the onboard segment can be augmented with more "intelligent" rules to maximize performance and minimize human operational attention required as a mission progresses and over a range of related missions. Careful design of such systems to enable them to be embedded and to be distributed is required, but this has been done successfully [8, 14].

Automatic Planning and Scheduling Automated planning and scheduling is incorporated in the ground segment simply to optimize the use of resources and schedule events based upon opportunities and events. Traditionally this function is done as a batch operation rather than a real-time function. Predetermined plans are then executed onboard as RTOS scheduled sequences. This system will include the capability to distribute planning between the ground and space segment such that automatic onboard replanning can enable a more autonomous spacecraft to meet higher level objectives in spite of system performance degradation (faults) and unexpected events. This requires a direct interface between the fault detection layer, the command and control layer, the RTOS scheduler, and the automatic planning system software. The extent to which the onboard planning system can replan based upon goal specifications is tailorable, but the ability to incorporate goals into plans, thus raising the level of abstraction of a plan from an explicit sequence to a more goal-oriented plan is to be standardized [17, 18]. Optimizing algorithms for resource management and scheduling must be "anytime" such that real-time operating system deadlines can be met. Finally, the automatic planning system must interface with the RTOS resource management kernel functions for management of standard resource types such as consumable resources, serially reusable exclusive access resources, etc.

Distributed Systems Internetworking Internetworking between space segment and ground segment computers will likely include parallel stacks at the ground entry point, but these parallel stacks should provide reliable data transport with transparent protocol conversion at the lowest layers possible. Furthermore, space communications protocols have special requirements based upon limited bandwidth and many other characteristics of space segment links not in common with ground segment networks and have traditionally only been point to point. However, the emergence of constellations, optical intersatellite links, and many other developments means that the need for space segment protocols with internetworking capabilities is more likely in future small satellites. Since the need for various standardized communications protocols is likely to continue to vary greatly for future small satellite systems, the proposed next generation small satellite operations system should include a message passing API which enables the use of different protocol stacks and the integration of interprocess communications within a heterogeneous system.

Currently, most mission operations teams are physically co-located and include constant monitoring. The automation framework proposed here will allow mission operations with management by exception. In order to actually realize cost savings associated with management by exception, yet still provide a system to support complex anomaly resolution, the system must include the capability to support distributed mission operations teams. Aside from the distributed computing environment described here which underpins this overall system, the ability to incorporate resources and individuals that are widely distributed is greatly enhanced with the ability to use public networks. The problem with public networks is security. Transparent software and/or hardware based security protocols can be incorporated into the data transport associated with command and control and data distribution such that public networks can be used. Data transport sub-layer encryption tied to system service client authentication is used to provide reasonable levels of security within a public network [15, 16].

Automation and Autonomy Testbed Automation must be carefully validated before it can lead to a more operable autonomous system. The careful integration of automated functions is considered to be fundamental to the achievement of operational autonomy. Without significant testing, automation may reduce performance and/or operability, actually leading to less autonomy and even more significant human attention requirements. In addition to traditional software engineering methods, a

development environment for simulations and mixed hardware/software testbedding with the ability to support injection of simulated faults, is fundamental to the successful application of automation. The use of a testbed to build a virtual spacecraft in order to be used as an automation testbed with the ability to migrate automation proven in the testbed will help ensure that automation does in fact lead to increased operability. For example, a fault detection and reaction automation can be prototyped, tuned, and tested in the testbed, then migrated to the ground operations computers for further testing and operator concurrence, followed by migration onboard once the automation is "trusted."

Data Management The final software function important to the overall operations concept is distributed ground and embedded system data management. Data stored within the distributed system of ground computers should be accessible from any distributed node. Further, data stored within the distributed space segment should be transported to the ground segment and stored at ground entry points as transparently and reliably as possible. Finally, the ability to distribute information, retrieve it transparently within a distributed system and to automate the retrieval of information most pertinent to management tasks is important. Often an understated problem of mission operations is simply the overhead of retrieving information required to support anomaly resolution, planning, and other human activities.

Prototype System

In order to test some of the concepts presented in this paper, a prototype system has been developed by students and faculty of the Colorado Space Grant College at the University of Colorado at Boulder. This prototype consists of an October 1997 manifested Space Shuttle Hitchhiker payload along with associated ground software and hardware.

Hardware Components The prototype includes equipment to be located at three different physical locations: the flight system to be located on the Space Shuttle, the ground support equipment to be located at Goddard Space Flight Center, and the central command equipment located at the Colorado Space Grant College.

A Sun SPARCstation 20 will serve as the ground support equipment and will be directly connected to NASA's communication channels. This machine will also serve as the primary ground system information archival unit by storing all down-linked data to a SCSI digital audio tape (DAT) drive. Remote connection support to the Colorado Space Grant College will be provided via an Internet connection and a backup high-speed modem link.

Sun SPARCstation 20, SPARCstation 10, and SGI machines will be used at the Colorado Space Grant College to command and control the payload and to process data received from the flight system.

The flight system consists of a Motorola IDP embedded development board utilizing a Motorola 68EC040 microprocessor, three science instruments, and a SCSI DAT drive which will provide primary flight system data archival.

The payload processor is responsible for controlling the three science instruments and for communicating with ground control through NASA communication links. During early prototyping, this processor communicates directly with the Sun SPARCstation 20 slated to perform the responsibilities of the ground control workstation to be located at Goddard during the Shuttle flight.

Software components The prototype include software in both the flight system and the ground system:

UNIX This non-real-time operating system is used as the common platform supporting the software packages utilized in the ground system.

RTEMS The Real-Time Executive for Military Systems is being used in the flight system to schedule processes and manage computer resources. RTEMS was developed by On-Line Applications Research Corporation under contract to the Research, Development, and Engineering Center of the U.S. Army Missile Command and is provided as a technology transfer/dual use service to individuals and companies having a need for this product. RTEMS is supported and is freely available via the Internet[1].

SCL The Spacecraft Command Language, a commercial command and control software package developed by Interface and Control Systems, is being used both in the payload and in the ground system. SCL scripts and functions can be used to perform certain activities. SCL constraints can be used to prevent certain commands from being executed when a condition exists in the system that could adversely affect mission success if the command is actually executed. SCL rules can be used to execute scripts and functions based on the current value of relevant SCL database items. Communication protocols can be defined in SCL so that the flight software can communicate with the subsystem hardware via custom

[1] http://lancelot.gcs.redstone.army.mil/rg4/rtems.html

drivers while the ground system software elements distributed between machines can communicate using standard techniques such as sockets.

Sammi This commercial graphical user interface (GUI) building tool is utilized to generate displays for the mission operations team. These displays are organized in a hierarchical fashion and allow a mission controller access to detailed information about subsystem performance and design. Color coding of display elements is used to signify system states (green=nominal, yellow=warning, red=critical).

SELMON This Selective Monitoring tool developed at JPL will be used to process sensor output and automatically trigger SCL rules based upon SELMON outputs, decision boundary logic, and contextual logic. SELMON will be used in both ground and flight systems.

The prototype system integrates detection methods with the SCL forward-chaining rule-based system so that fault alarms and detected opportunities can be linked to intelligent reactions. The integration of SELMON monitoring with SCL has been implemented both with shared memory and message passing such that this monitor-to-action linking can be used locally or for actions to be taken anywhere within a distributed system.

Plan-It II A ground system planning and scheduling tool developed at JPL, Plan-It II will be used to aide in the mission sequencing. Current development efforts include modifying Plan-It II to allow automatic generation of SCL scripts, rules, and constraints.

TCP/IP The Transport Control Protocol layered on the Internet Protocol provides reliable data transport between the computers comprising the ground system.

Kerberos Security is incorporated in the networking software by utilizing the Kerberos package developed at MIT.

LabVIEW Laboratory Virtual Instrumentation Engineering Workbench is a commercial tool which is being used to simulate the spacecraft subsystems during initial development and will also be used during flight to test software on the ground before it is migrated to the flight system. As each subsystem design progresses, its functionality is imitated in LabVIEW. Utilizing a sockets connection, SCL queries LabVIEW for the current state of each subsystem, much like it will be utilized to query the real instruments during flight. As each subsystem is prototyped, its hardware replaces its respective LabVIEW simulation, until at last the complete spacecraft is integrated. During flight, the ground system software will test new command software using the LabVIEW hardware simulation. Once the code is validated, it will then be migrated to the embedded system on the shuttle.

O2 This commercial object oriented database is being used to maintain a history of SCL database values, to manage the configuration of source code and data, as well as to store design and operations documents.

HTML The HyperText Markup Language is being used to create World Wide Web pages[2] which document prototype design details. This information is used during the development stage to distribute design details between subsystem engineers, and will be used by mission operations personnel during training and for reference during the flight. Web browsers such as Netscape and Mosaic are currently being used to access these Web pages.

C/C++ Device drivers, networking and test code are being developed in C and C++ and are compiled using the freely available gcc and g++ compilers.

Conclusions

Concepts for an architecture providing a means for controlling a small spacecraft using a minimum of personnel have been discussed. By utilizing reusable hardware and software, industry standards, and software which enhances autonomous operation, the "smaller, faster, cheaper" philosophy is adhered to, thus providing an architectural framework around which future designs may be molded. With adequate flight system processing power and software design, it should be possible to improve spacecraft performance while simultaneously reducing the number of mission operations support personnel by at least an order of magnitude. The architecture will be demonstrated and the operations personnel reduction hypothesis tested on an October 1997 Space Shuttle Hitchhiker payload.

References

[1] James Stuart, "New Approaches to Commercial Space Communications Systems", AIAA 93-4244, (Huntsville, Alabama, September 1993).

[2] Michèle Basseville and Igor Nikiforov, Detection of Abrupt Changes: Theory and Application, Prentice Hall: Englewood Cliffs, NJ, 1993.

[3] R. Bonometti, "The Role of Small Satellites In The Emerging National Information Infrastructure And

[2] http://www-sgc.colorado.edu

The Future Defense Global Grid", Advanced Research Projects Agency, 1993.

[4] Richard Doyle, Steve Chien, U. Fayyad and E. J. Wyatt, "Focused Real-Time Systems Monitoring Based on Multiple Anomaly Models," unpublished manuscript, Artificial Intelligence Group, Jet Propulsion Laboratory, 1992.

[5] Richard Doyle, "A Distance Measure for Attention Focusing and Anomaly Detection in Systems Monitoring", unpublished manuscript, Artificial Intelligence Group, Jet Propulsion Laboratory, 1994.

[6] Ewald Heer and Henry Lum, "Toward Intelligent Robot Systems in Aerospace", American Institute of Aeronautics and Astronautics, Inc., 1988.

[7] S. Kullback, J. Keegel and J. Kullback, Topics in Statistical Information Theory, Springer-Verlag: N.Y., 1987.

[8] C.J. Paul, Anurag Acharya et al., "Reducing Problem-Solving Variance to Improve Predictability," Communications of the ACM, Vol. 34, No. 8, August 1991.

[9] H. Tokuda and C. Mercer, "ARTS: A Distributed Real-Time Kernel", ACM Operating Systems Review, Vol. 23, No. 3, July 1989.

[10] Kristin Bruno, Linda Welz et al., "Analyzing Human Errors in Flight Mission Operations," unpublished manuscript, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1991.

[11] Robert M. Manning, "Low Cost Spacecraft Computers: Oxymoron or Future Trend?," Jet Propulsion Laboratory, Paper AAS 93-031, 16^{th} Annual AAS Guidance and Control Conference, Keystone, CO, February 1993.

[12] F. W. Sexton, D. M. Fleetwood, et al., "Qualifying Commercial ICs for Space Total-Dose Environments," IEEE Transactions on Nuclear Science, Vol. 39, No. 6, December 1992.

[13] D. M. Fleetwood, P. S. Winokur, et al., "Hardness Assurance for Low-Dose Space Applications," IEEE Transactions on Nuclear Science, Vol. 38, No. 6, December 1991.

[14] Brian Buckley and Louis Wheatcraft, "Spacecraft Command Language – A Smart Control System," Interface and Control Systems, Melbourne, Florida, Barrios Technology, Houston, Texas, March 1991.

[15] John Ioannidis and Matt Blaze, "The Architecture and Implementation of Network-Layer Security Under Unix," unpublished manuscript, Columbia University and AT&T Bell Laboratories, 1993.

[16] David Borman (ed.), "Telnet authentication: Kerberos version 4.", RFC 1411, 1993.

[17] Pattie Maes, "Situated Agents Can Have Goals," Robotics and Autonomous Systems, Vol. 6, 1990.

[18] Leslie Kaelbling and Stanley Rosenschein, "Action and Planning in Embedded Agents," Robotics and Autonomous Systems, Vol. 6, 1990.

[19] Robert L. Staehle, Elaine Hansen, et al., "Pluto Mission Progress: Incorporating Advanced Technology," 7^{th} Annual AIAA/Utah State University Conference on Small Satellites, Logan, Utah, September 1993.

[20] Elaine Hansen, "Lowering the Costs of Satellite Operations: Lessons Learned from the Solar Mesosphere Explorer (SME) Mission," University of Colorado, Paper AIAA-88-0549, AIAA 26th Aerospace Sciences Meeting, Reno, Nevada, January 1988.