A Successful Component Architecture for Interoperable and Evolvable Ground Data Systems

Tracking Number: 58308

Danford S. Smith*, John O. Bristow†, Jonathan J. Wilmot‡
National Aeronautics and Space Administration, Greenbelt, Maryland, 20771

The National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC) has adopted an open architecture approach for satellite control centers and is now realizing benefits beyond those originally envisioned. The Goddard Mission Services Evolution Center (GMSEC) architecture utilizes standardized interfaces and a middleware software bus to allow functional components to be easily integrated. This paper presents the GMSEC architectural goals and concepts, the capabilities enabled and the benefits realized by adopting this framework approach. NASA experiences with applying the GMSEC architecture on multiple missions are discussed. The paper concludes with a summary of lessons learned, future directions for GMSEC and the possible applications beyond NASA GSFC.

Nomenclature

API	=	Application Programming Interface
CAT	=	Criteria Action Table. GMSEC Automation Component
COTS	=	Commercial Off-the-Shelf Software
GMSEC	=	Goddard Mission Services Evolution Center
GREAT	=	GMSEC Reusable Events Analysis Toolkit
GSFC	=	Goddard Space Flight Center
MOC	=	Mission Operations Center
MOM	=	Message-Oriented Middleware
NASA	=	National Aeronautics and Space Administration
SMEX	=	NASA's series of Small Explorer Satellites
ST5	=	NASA's Space-Technology 5 constellation of three satellites
TRMM	=	NASA's Tropical Rainforest Measuring Mission

I. Introduction

The Goddard Mission Services Evolution Center (GMSEC) reference architecture has been in development since 2001 and operational since mid-2005. GMSEC is NASA Goddard Space Flight Center's initiative to move towards a common framework open architecture with the hopes of decreasing development, integration and operations costs while increasing system flexibility and capabilities.

* Manager, Goddard Mission Services Evolution Center, NASA Goddard Space Flight Center, Information Systems Division, MS 581.0, Greenbelt Road, Greenbelt, MD USA 20770
† Technical Lead, Goddard Mission Services Evolution Center, NASA Goddard Space Flight Center, Information Systems Division, MS 581.0, Greenbelt Road, Greenbelt, MD USA 20770
‡ Task Lead, Advanced Flight Software Architectures, NASA Goddard Space Flight Center, Information Systems Division, MS 581.0, Greenbelt Road, Greenbelt, MD USA 20770

A. NASA Goddard Space Flight Center

The Goddard Space Flight Center (GSFC) in Greenbelt, Maryland operates most of NASA's sub-orbital and low-earth orbit unmanned scientific spacecraft. GSFC manages a wide variety of missions, from balloon and sounding rocket experiments to flagship missions like the Hubble Space Telescope. Mission durations for orbital spacecraft range from several months to over 20 years.

About 30 satellites are managed by GSFC at any time, with about half of these operated from mission operations centers (MOCs) on the Greenbelt campus. Others are managed by universities across the United States. Suborbital missions are managed from Wallops Flight Facility in Virginia.

B. The Need for a New Ground System Approach

Traditionally, each satellite mission team at GSFC is responsible for developing its own mission operations center (MOC). Typically, the ground system manager utilizes the same designs, concepts and approaches used on other recent missions; usually the last mission he or she worked. Trade studies are done, comparing different key components, with heavy weighting given to the most familiar.

Although successful, this approach has led to several shortcomings:

1. **Innovation is slowed.** Mission managers are risk adverse, and budget-constrained missions work towards low-cost solutions, only making enhancements when absolutely necessary. New capabilities, developed to address issues on a single mission, often cannot be applied to other missions because the lack of commonality in the developed systems.

2. **Maintenance costs are high.** There are cases where multiple in-house software options exist for the same domain area; each used supporting multiple missions. With costs constrained on maintenance support, valuable enhancement money is split among products. As a result, products tend to get old, with little or no significant enhancement over time. The underlying architecture of these systems is based on approaches from the late 1980s and early 1990s.

3. **COTS products are rarely used.** Often, COTS product lines are not even considered because the products have never integrated with other GSFC products. Those performing the studies are not familiar with the COTS products and tend not to trust the "unknown".

4. **Efforts are duplicated.** Studies, designs and implementations efforts are repeated, although very similar, for each mission.

A key problem facing all of the missions today is developing and operating within reduced budget guidelines. Not only is the traditional development approach considered expensive, but the lack of innovation is felt to restrict the introduction of new concepts which can lower operations costs over the life of the mission.

C. Goals for a New Architecture

Goals were established prior to starting work on the new architecture. It was important that any new investment specifically address the issues identified by the GSFC management and mission teams. Candidate solutions would be measured against the goals to ensure progress was maintained. By keeping the list of goals short, presentations to management and new missions could be kept to a high level and not be overburdened by technical details. The following four key goals for GMSEC were developed early in the program and have continued to drive decisions throughout the system's development:

1. **Simplify integration and development.** The architecture should allow for reduced system integration time, shortening overall development efforts. Clean interfaces and clear functional boundaries for components are key to simplifying integration. A loose coupling of components and black box approach are necessary. Although GSFC has many heritage components, they often have to be significantly modified to be used on the next mission with unique interfaces and integration becoming a long custom process.

2. **Facilitate technology infusion over time**. Many GSFC missions are develop 3-4 years prior to launch and operated for over a decade. Over such a time span, technology evolves, operations concepts grow, security issues and patches arise, and new components and tools become available. This model requires full system change-out about every five or so years. The GMSEC architecture must expect and facilitate routine changes over the life of the mission.

3. **Support evolving operations concepts.** The new architecture should enable new operational concepts now envisioned and should not preclude those to come. Key new concepts include fleet operations, satellite constellations, split operations between GSFC and other organizations and ground-based or on-board autonomy.

4. **Allow for a mix of government, heritage, COTS and new components**. GSFC has made a great investment in custom mission support software. The new architecture must be able to utilize the existing systems while allowing new components to be added. The architecture should easily allow the use of the commercial products – with the assumption that a commercial product's actual software can not be changed. In addition, through utility routines or simple interfaces and standards, the ability to write and integrate new applications must be straight forward.

II. The GMSEC Architecture

A. High-Level Architecture Concepts

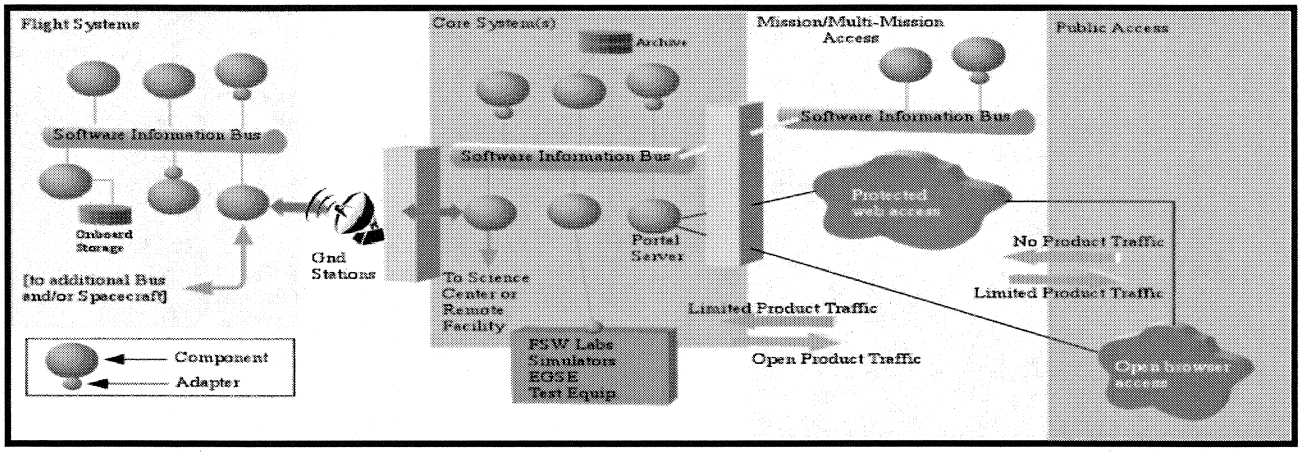
As with the list of goals, the list of high-level architectural concepts for GMSEC has been kept short:

1. **Standardize interfaces** – not Components. GMSEC does not perform trade studies and make component recommendations or selections. By standardizing interfaces, GMSEC encourages the access to a broad range of tools.

2. **Middleware infrastructure.** At the heart of the architecture is a message oriented middleware (MOM). It can be a commercial package, open source or GSFC-developed.

3. **User choice.** GMSEC is not trying to select the best component in each functional area. The GMSEC team is not trying to compare COTS products against each other or against heritage systems. Instead, the architecture allows the user or mission to select their favorite tools and integrate them into a ground system.

4. **General-purpose approach with flight-ground capabilities.** The architecture itself should be designed to be adaptable to any number of missions. The key concepts should be applicable to either flight software or ground control systems and should, ideally, be extensible to other domains.

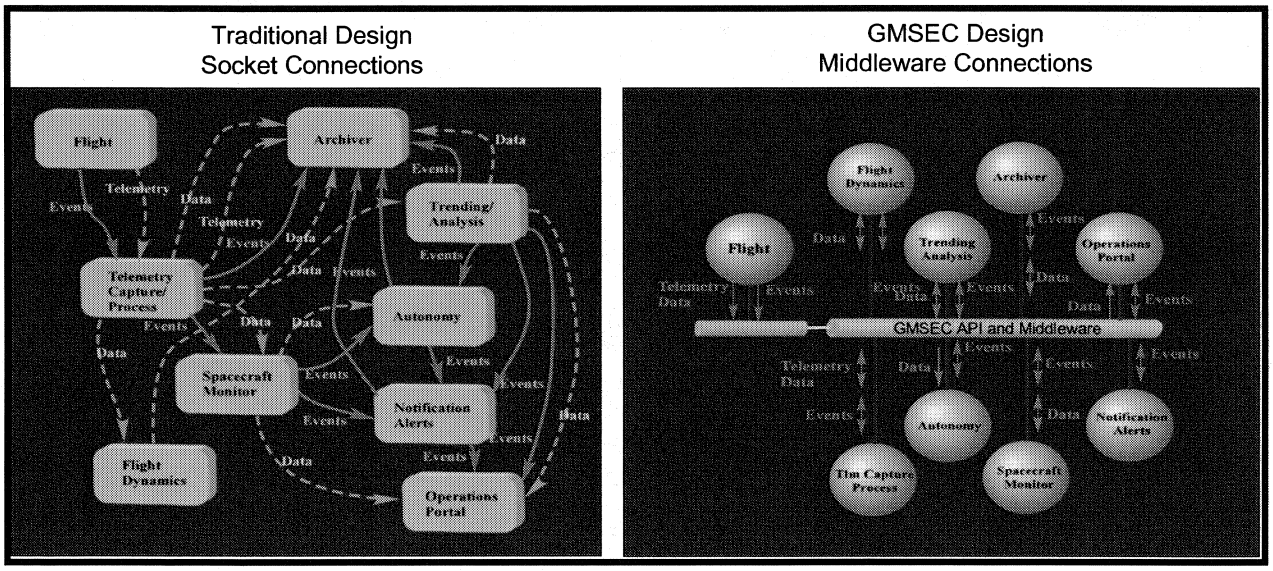


The basic GMSEC architecture concepts are applicable to flight systems and ground systems, and can be extended through firewalls to provide external access.

B. The Message-Bus Approach

The GMSEC Architecture uses a message bus (sometimes called an information bus or software bus) for inter-process and inter-node communication. Instead of traditional socket connections among components, each component only interfaces with the message bus. The middleware keeps track of where processes are located and which process requires the data published to the bus.

The message bus provides publish/subscribe message passing mechanisms. Applications "publish" messages to the bus. Each message contains a subject name and the standard message contents. The subject name, for GMSEC applications, indicates the mission, originating node, type of message, etc. Applications that need the data "subscribe" to the pertinent subject name(s) and the middleware delivers the messages which match the subscribe request.



The use of a publish/subscribe middleware package reduces the complexity of system interfaces and simplifies component integration.

Although publish/subscribe mechanisms are common to many different middleware products, each product uses its own proprietary message structure for passing the data on the bus. The commercial middleware products are

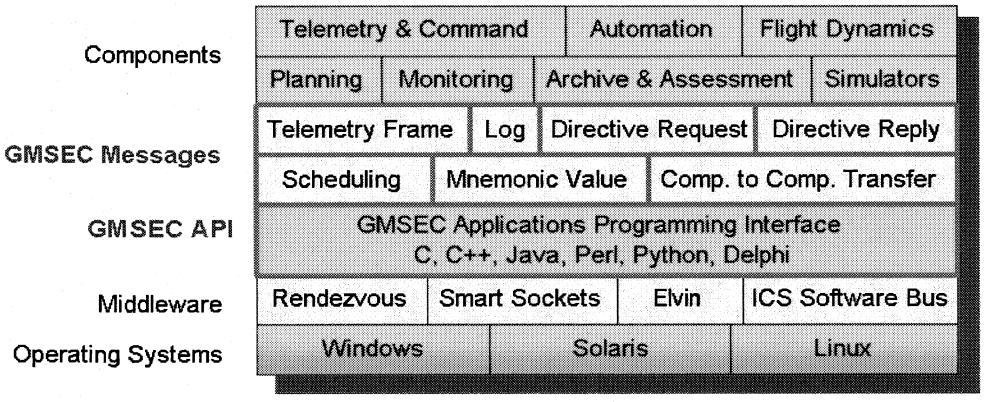
therefore not compatible with each other and applications are normally written to match the specific middleware package selected for the system development effort. The GMSEC API (see next section) normalizes the basic capabilities of multiple middleware products so they each appear the same to the applications software. In this way, a change to the middleware product does not require changes to the applications software.

The design concept of normalizing the middleware packages and creating a common API was a result of wanting to avoid vendor-lock-in. It may be necessary to change a middleware product if a company goes out of business, the product goes in a new technical direction, product costs increase dramatically or other problems are found. Another significant benefit is the ability to use different middleware packages for different situations. A no-cost GMSEC-developed middleware system can be used during development efforts and a more robust, higher capacity system used for final integration or operations. For the highest reliability, GMSEC supports the use of commercial middleware packages that were initially developed to support the U.S. banking industry and Wall Street. A messaging system with a very small memory footprint may be necessary for flight software. A product or new component can be tested with one middleware system and can then be expected to run under the other systems in different environments. The API also supports mixing middleware products in the same MOC.

Plug-in tools have been developed by the GMSEC team to allow monitoring of a system or bus configuration, resource utilization or bus traffic, and to automate middleware failover scenarios. A full performance analysis has been performed and real-time monitoring tools have been developed. The GMSEC system, with its middleware, can transfer 10's of megabits of data per second – a rate 100's of time beyond what missions today typically require for command and control and health and safety data. Typical overhead of the GMSEC middleware is less than 2% of the CPU.

C. API and Message Standards

The GMSEC API forms a critical layer in the GMSEC architecture. It provides isolation between the applications programs and the underlying messaging software. As discussed in the previous section, any of several different middleware packages can be used without modifying the applications. In addition, the API supports multiple languages, operating systems and platforms. It normalizes the behavior of the middleware while allowing access to special functions or capabilities of individual middleware products.



The GMSEC API provides the generalized interface between applications programs and the system middleware. Multiple languages, middleware products, platforms and operating systems are supported.

GMSEC standard messages are passed to the API by the applications. The GMSEC message specifications cover common messages such as telemetry packets and frames, event/log messages, heartbeat messages, directives, products, etc. Each message contains a subject naming standard, a common message header and the message-specific body. The naming standard is used by the middleware to route the messages. The common header contains more information about the message (time tag, etc.) and may be used for further applications-level filtering and

processing decisions. The message body is tailored for each message type. To develop the message standards, the GMSEC team utilized the interface documents from many different commercial product vendors as well as the documentation on the GSFC-built systems and the experiences of many of the mission development managers. The result has been a message set which has been very easy for many product vendors or developers to adapt to.

The GMSEC API is key to rapid and successful development efforts and was designed with usability in mind. It is software designed for software people. The documentation has been refined by having fresh-out employees install and use the API without help – if they had a question, it was addressed as an update to the documentation. For L3 Communications, the result was COTS software that worked the first time they brought it to the GMSEC Lab; same for Attention Software, who relied solely on the documentation and developers toolkit and did not make a single call for technical support.

Because of the significant number of platforms, operating systems and programming languages the GMSEC Architecture and API must support, an extensive automated testing system was developed. The test suite contains over 12,000 tests that are automatically executed daily as new code is check into the configuration library. This test suite was absolutely essential for supporting so many platforms and languages. Since it runs every day, errors are detected as soon as a new routine is checked in and can be isolated and corrected quickly.

Together, the GMSEC API and standardized messages provide a plug-and-play capability for a wide variety of functional components across a large selection of hardware platforms, operating systems and development languages. By involving the commercial product industry in the message definition process, the message standards were designed to support a wide range of products, concepts and problem domains. The success of the message definitions allowed simplified integration of COTS products as well as heritage GSFC software systems and software from other NASA Centers.

D. Compliant Components

Instead of selecting the "best in class" components for telemetry and command processing, trending, planning and scheduling, etc., GMSEC uses the common interface approach to allow many different products of the same functional domain to be integrated. By having choices in each functional area, missions avoid vendor lock-in and can select components based on merits (technical, cost, etc.). There is no single suite of tools to define a GSFC GMSEC control center. Flexibility is maintained.

Components on the GMSEC framework provide the system's functionality. Components can be as major as telemetry and command systems or planning systems or as small as performance monitoring tools or system agents.

Each major component is required to meet certain standards to be considered "GMSEC compliant":

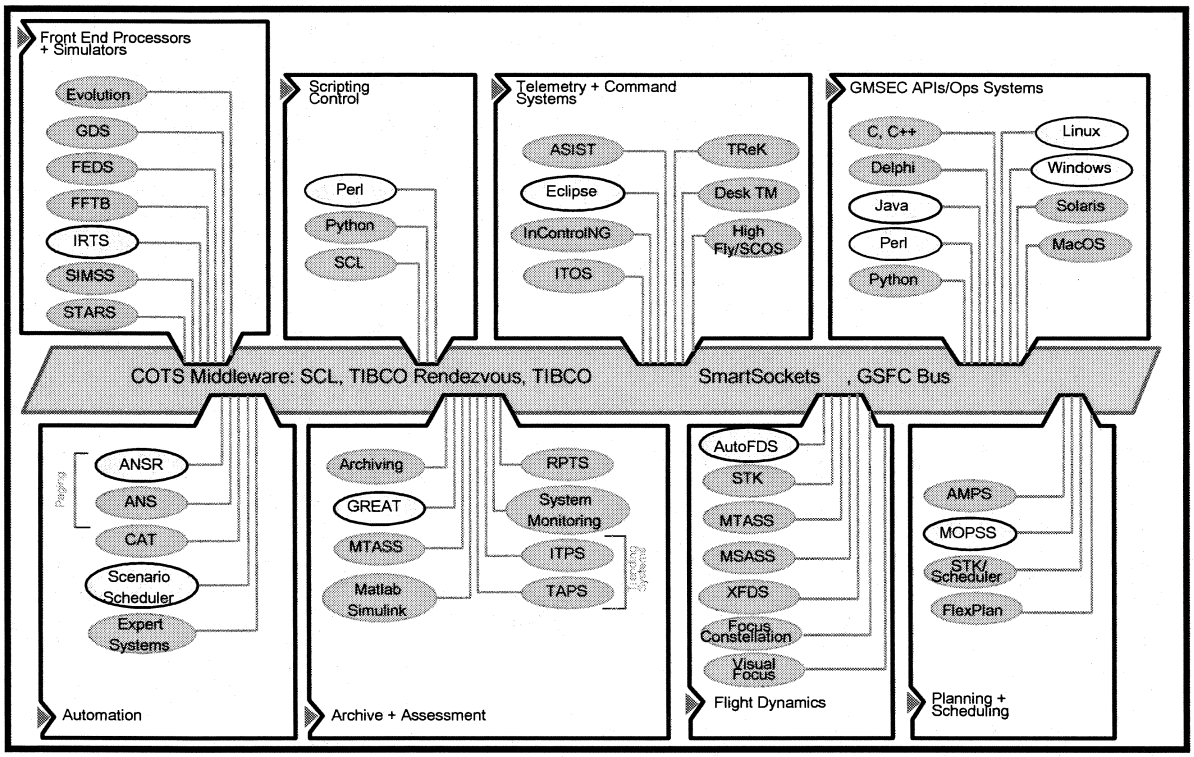
1. It must, of course, meet its functional requirements;
2. It must publish a heartbeat message on a periodic basis;
3. It must publish status/log messages to indicate an action has taken place or an event has occurred;
4. It must support user directives for component control to be received over the message bus.

These simple rules yield very powerful results. The heartbeats allow for system monitoring, configuration displays and failovers. The log messages and directives are required for system automation (see section III).

Source code for most COTS products can not be altered to become GMSEC compliant. An "adapter" approach is used for these components. An adapter is a piece of software which works like an API-to-API interface and converts from the COTS package's interfaces to the GMSEC interfaces. Because the GMSEC interfaces were developed with knowledge of many COTS interface definitions, this adaptation has proven to go quickly (from as short as a day to about 2 weeks). COTS packages that do not have clean, exposed APIs are more difficult to adapt and their underlying design may not be amenable to GMSEC adaptation. Obviously, new software can be written to use the GMSEC API calls directly. Heritage software can use either the adaptation model or the new software model.

The goal of matching the interfaces is to create components that can be instantly integrated when connected to a GMSEC bus. By significantly reducing the integration step, efforts can begin immediately on tool-specific configuration to support the mission's needs. Instead of "plug-and-play", it should be "configure-and-operate."

GMSEC does not advocate on behalf of specific components. Instead, the GMSEC organizations serves as an honest broker – encouraging the adaptation of many components and leaving the responsibility for product selection to the end-user. GMSEC now offers choices in each of several functional areas. The first GMSEC components were programs already in use at GSFC that could be modified to use the GMSEC API and help validate the plug-and-play concepts to refine the message specifications. Commercial vendors' products were then added. Components enabled by the architecture, including performance monitoring tools and automation suites, were developed by the GMSEC development team. The result is a robust "catalog" of choices in several functional areas. The catalog is added to on a regular basis with additional heritage, newly developed, and commercial products.



The GMSEC "catalog" is shown above – the acronyms are not as important as understanding that they represent options for the mission end-users.

E. Architecture Analysis and Comments

Commercial product vendors and mission development teams have reported that the GMSEC API is very well documented. The developer's toolkit allows for quick use and integration including providing templates for all message types. Attention Software, Inc. was able to modify and integrate their paging system with the GMSEC message bus simply by following the documentation. They did not call the GMSEC team until they were ready to demonstrate their product and the demonstration worked perfectly on the first try. When L3 Communications integrated their telemetry and command product, In Control Next Generation (ICNG), they first spent two weeks working at their facility. When they installed their product in the GMSEC Lab, it was publishing GMSEC messages visible on the GMSEC monitoring displays before the ICNG user interface finished starting up.

GMSEC is considered an unusual piece of software by some. Instead of being a solution for a specific functional need, GMSEC addresses system-wide issues of flexibility, scalability, maintainability, technology infusion over time, avoidance of vendor lock-in, reuse, and COTS integration. By addressing these key issues, GMSEC enables mission development teams to develop systems more rapidly from a large catalog of parts. Vendor lock-in issues are minimized and component changes can be made without impacting with existing components – a tremendous advantage over traditional maintenance approaches.

New trends are now being seen with the missions using the GMSEC architecture. Multiple choices of components allow technical needs or even personal preferences to factor into selections. The first three missions each chose a different telemetry and command system. At the same time, tools that perform very well will begin to be used by many missions, and the number of available choices may actually shrink in some domain areas over time. This has already been experienced in areas such as paging systems.

III. Architecture-Enabled Capabilities

The early demonstrations of the GMSEC system displayed the message bus concepts and showed the simplified integration of heritage and COTS products. These demonstrations, however, did not show the functional benefits of the architecture. In fact, it first appeared that GMSEC was a more expensive approach to providing the same mix of functional components that had been available for years. What was not yet apparent was that the architecture itself, through the use of standard messages and key requirements placed on the components allow for significant increases in capability and reliability over previous approaches.

A. Situational Awareness, System-Wide Control, Automation and Failovers

Event messages (often called log or status messages) have been traditionally available in displays or log files associated with many different components. Although the GMSEC approach does not preclude the local logs, the publishing of log messages onto the bus is required. The message bus, therefore, is carrying the combined messages from all components in all functional domains.

Simple tools can be developed which display or filter the combined collection of messages from all the subsystems. This provides **situational awareness,** allowing users to view messages that span multiple domains in one place. The GMSEC Reusable Events Analysis Toolkit (GREAT) is a compliant utility that provides the viewing and filtering of messages. GREAT displays can be configured by the user to include selected message fields, sort and filter the messages. In addition, GREAT creates a merged data base of the messages and has powerful report generation capabilities which operate off of the archive.

This enhanced situational awareness can significantly increase the effectiveness of the flight operations team. Take for example a system where scheduling, command management, flight dynamics and real-time telemetry and command events are logged together. The real-time system could report that a temperature sensor is out of limits low. In a traditional architecture, the user may then plot the value or look elsewhere for the cause. With situational awareness and the GREAT display, the operator could tell that there are five minutes left in the pass (scheduling message), a stored command on-board turned a heater off two minutes earlier (command management), and that the satellite had entered into eclipse about one minute earlier (flight dynamics). On one display, the operator could quickly surmise that the heater should have been turned on, not off and there are several minutes left in the pass to correct the problem before it gets worse.

Requiring each component to subscribe to text-based directives (scripted or user inputs) on the bus provides **system-wide control**. Scripts of directives destined for any number of applications can be executed from a single location. An entire demonstration or operation can be conducted from one system.

With situational awareness and system-wide control, the user has the basic ingredients for **automation**. An

application can be developed which watches messages on the bus (situational awareness), makes decisions based on those messages (new capability), and takes action (system-wide control) per the rules an operator has established. GMSEC's Criteria Action Table, "CAT", allows users to develop automation rules. The rules indicate the "criteria" for an action to be taken and the series of steps to be taken if the conditions are met. Any message can be watched for and parsed. Multiple message combinations across multiple domains, temporal criteria and local variables may be included. The actions taken can be defined as a sequence of functions, including relative timing, parameter setting, operating system directives, message generation, user control directives and temporary rule disabling.

The GMSEC CAT system allows for event-driven system automation. It enables known actions to be taken in known conditions. Model-based automation is possible by having independent model-based reasoning tools make determinations and publish event messages for CAT to act on, or by issuing its own directives.

The first teams of GMSEC users found a broad set of uses for the CAT automation. A simple rule could be to watch for the end of a pass, then wait 60 seconds and close files, trend critical data and distribute products to external organizations. If a pass does not start when planned, the tool could invoke a reacquisition sequence. In conjunction with a schedule execution system, total operations can be automated for several shifts per day and users can be paged (or other actions taken) when anomalous conditions are recognized.

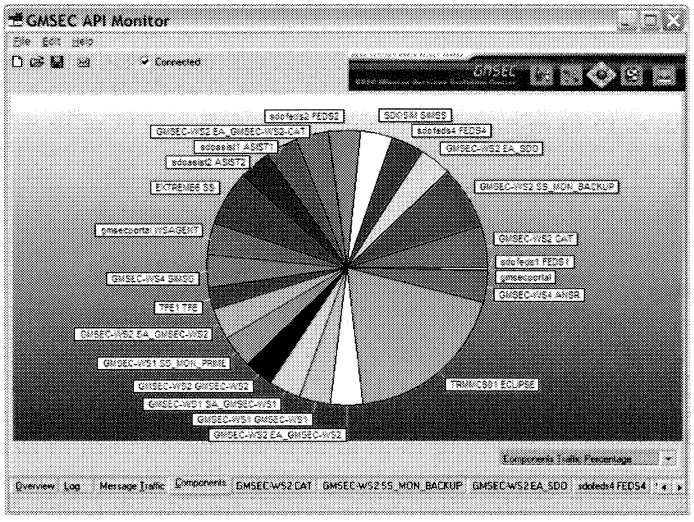
The GMSEC SystemAgent is a general purpose GMSEC-compliant software component that provides among other things, software component or computer health information to other GMSEC components such as CAT on the message bus. It provides a mechanism for managing failures of the middleware or individual components without human intervention. It can even help predict failures by monitoring the computer's available memory or disk space. Other failures may be detected by lost or missing heartbeat messages. CAT rules and SystemAgents may used to watch for these problems and take action. Should a heartbeat for a given component not be received during a set period, CAT in combination with the SystemAgent can direct the application to restart either on the same machine or on another machine. The message bus middleware automatically reestablishes communications paths. If a node goes down, or all of its applications, then all of its applications can be automatically started on another machine or the applications can be split across multiple machines. Failover rules can now be set up in a number of hours. The flexibility of application- and node-level failovers offer tremendous advantages over the older full string failover approaches.

B. Creative Tools

Having data moving across the bus creates opportunities for tools to be developed which monitor the system itself. Components can be developed which subscribe to certain message types and do not need knowledge of the major functional pieces of the system.

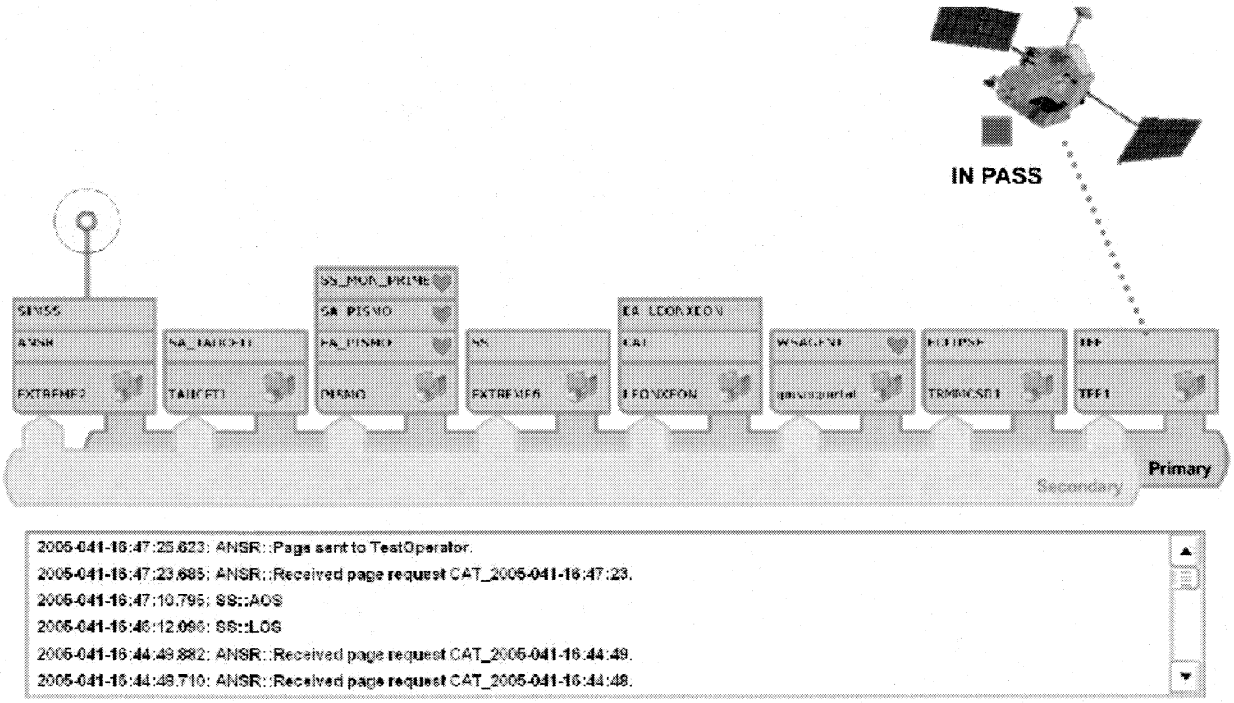
One simple tool subscribes to all message types. It then tabulates the number and sizes of messages for each type and creates a pie-chart of current bus traffic. Users can watch the traffic build as telemetry streams are started or major activities take place and can possibly be aided in problem identification.

Another tool records all message traffic for a period of time and then allows its playback at a later time. By moving the data file to a laptop, realistic demonstrations of actual message traffic and component processing can be prepared to show specific scenarios. For repeatable problems, this message recorder can be used to track down issues without impacting an operational environment.



The display below shows the dynamic configuration display. By monitoring heartbeat messages and defining automation rules in CAT, a display of the actual system configuration can be created. Each "stack" represents a single node on the network. Each box in the stack is a

software component for which a heartbeat is being received. Alarms go off and colors change if the heartbeats stop for a component or components. Additional messages drive other parts of the display. When a start-of-pass message is seen, the satellite picture is drawn and a line drawn to the front-end –processor. When a message is seen saying someone is being paged (normally due to a problem), then the display is updated to show a pop-up antenna, an alarm is sounded, and the message is added to the high-level message area. Other messages can cause other updates to be made on the display. The importance of the display is that it represents the "'truth model" of the state of the current system – it is not simply a depiction of what was planned or anticipated.



The GMSEC configuration and activity display is created by passively monitoring traffic on the bus – there is no integration with the primary functional components.

C. Satellite Fleet and Constellation Operations

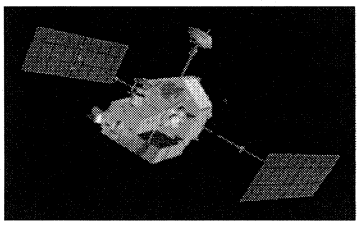
The GMSEC approach is ideal for the support of satellite fleets or constellations. Each satellite-related message on the bus includes a Satellite-ID field in the header. Telemetry data from multiple satellites can be put on the bus. A multi-satellite application can subscribe to messages containing any of several Satellite-IDs. Applications that only support a single satellite only need to subscribe to that satellite. Situational awareness tools like GREAT and automation tools like CAT can be configured to display or support multiple satellites. Multi-satellite displays are easy to create. Cross-mission collaboration tools can be developed by using the status from one satellite to determine actions to be taken on another.

Even with just a single satellite, data on the bus can be routed, for example, to multiple telemetry and command systems or to current and updated components for comparison testing or parallel operations.

IV. Successful Mission Use

The GMSEC Architecture is now operational on several mission ground systems including NASA/GSFC's Tropical Rainforest Measuring Mission (TRMM), TRACE, SWAS, WIRE and Space Technology 5 (ST5) missions.

A. TRMM



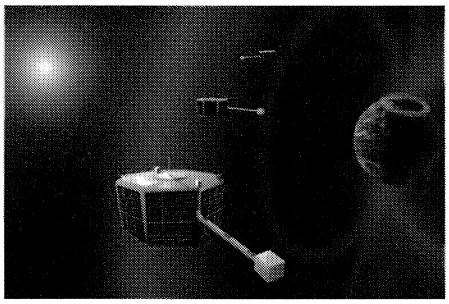
TRMM was the first mission to apply the GMSEC technologies. The TRMM spacecraft was performing well, but was beyond its budgeted lifetime. The Project was challenged to reduce operations costs by 50% or turn off the spacecraft. By basing their reengineering effort around the GMSEC Architecture and concepts, the TRMM mission was able to add significant levels of automation and move to fewer operational shifts per day without any science data loss. The annual operating budget was reduced by about 50% and the reengineering efforts were fully paid for out of incremental budget savings. The reengineered system has paid for itself within 2 years.

B. NASA Small Explorer (SMEX) Missions

The Small Explorer missions, including the TRACE, SWAS and Wire satellites did not implement the GMSEC architecture to save money, but rather to enable new capabilities associated with satellite fleet operations and constellation operations. The system has been operational since mid-2005 and has successfully demonstrated continuous "lights-out" operations. No direct operator involvement in the control room is required – the system pages users if a problem is detected.

The SMEX reengineering effort is considered a pathfinder for future GSFC low-cost fleet operations and the operations involving of satellite constellations.

C. Space Technology 5 (ST5)



The ST5 mission is a constellation of three small satellites launched in March 2006. Its control center was used to support satellite integration, pre-launch checkout, training, and now on-orbit operations. Power and solid-state recorder subsystems were modeled using a Matlab/Simulink GMSEC-compliant component. Real-time telemetry is used to update the model so that predictions can be made based on scheduled future activities. Should problems be projected, the modeling system notifies the scheduling system to make adjustments. All of the interactions are across the GMSEC bus.

ST5 has reported that GMSEC saved them money while allowing them to demonstrate more advanced capabilities, but the savings have not been quantified. ST-5 has also demonstrated limited "lights out" automation with GMSEC and is preparing for a full 2-week lights-out operations period with the system paging the operations team if problems are recognized.

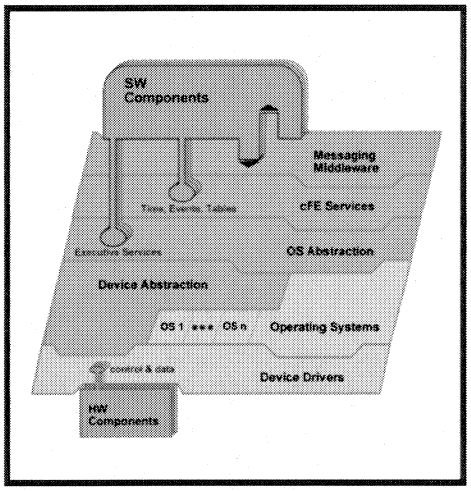
V. Potential Applications

The core GMSEC system, including the architecture, middleware and API has been designed to be domain independent and, therefore, applicable well beyond the confines of GSFC. It should be noted that the use of a message bus is a common software development strategy for large complex systems today. Most often, however, large systems are designed as "point solutions", with a vendor's product at the core and specialized interfaces designed to simplify the specific system's development efforts. GMSEC has taken these industry concepts beyond the "point solution" paradigm and has created a general purpose framework where the platforms, operating systems, development languages or even the underlying messaging mechanism can be mixed or changed without impacting the integrity of the overall system. It is this extension of the commonly used design approach which has drawn so much interest to GMSEC. GMSEC addresses concerns of those developing systems with very long lifetimes or a significant mix of computing resources and products.

The initial application of the GMSEC architecture for reengineering efforts of existing missions at GSFC has proven the value of the message bus approach. Other missions at GSFC which are being reengineered include

TERRA, AQUA, and AURA – the earth sciences trio basing their systems on the TRMM implementation. Future NASA missions now working with the GMSEC team include: GLAST, SDO, GPM and MMS.

GMSEC labs have been established at most other NASA Centers for evaluation. As a result of work at the Marshal Space Flight Center (MSFC), major software components have been traded between the Centers. The NASA Exploration Initiative planning teams are aware of GMSEC and studying its features for possible use on NASA's future manned missions and lunar and Martian exploration missions.



Other commercial space operators and other U.S. Civil government organizations have also been in contact with the GMSEC team about possible applications.

Many of the same concepts proven with GMSEC in recent ground system implementations can also provide benefits for flight software systems. GSFC has recently developed a small-footprint architecture for flight. This system includes device and operating system abstractions, a messaging middleware and a suite of core flight executive (cFE) services. The flight architecture has been tested on an existing experimental satellite and is planned as the primary architecture on a GSFC mission now under development.

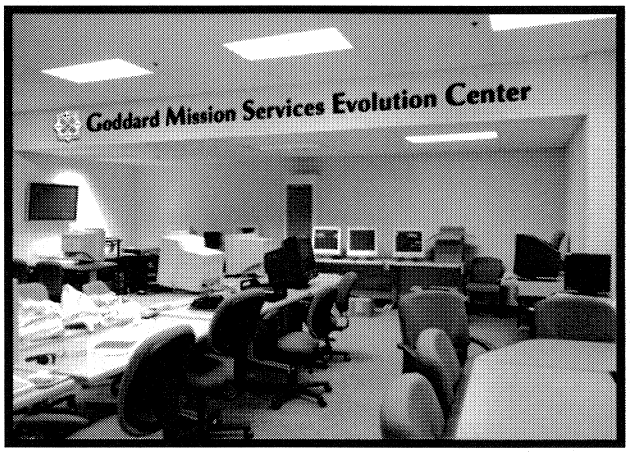
VI. Lessons Learned and Future Directions

The GMSEC project at NASA/GSFC has matured into a proven, successful benefiting multiple on-orbit missions and planned for several more. The GMSEC reference architecture approach, from technical to business model, has resulted in several key findings. The benefits that have been observed include the following:

1. Significant reduction in integration time
2. Components can be added/upgraded without impacting the existing system
3. Ideal for using multiple small distributed development teams and vendors
4. New concepts emerging for small independent components that integrate with the bus and provide immediate benefits
5. Missions are more willing to adopt the approach if "old favorite" components can still be used
6. Some vendors see message compliance as a way to enter what had appeared to be a closed marketplace
7. Standard message approach provides collaboration possibilities with other organizations
8. Same concepts can be extended to flight systems

In evaluating the actual GMSEC effort, a different type of self-assessment emerges (in no particular order):

1. Our interface focused approach was key to customer buy-in, allowing them to use their "old favorites" while saving integration costs.
2. The establishment of a well-equipped GMSEC Lab for integration and demonstrations provided a integration, development and testing show place.
3. A working relationship with industry has yielded new opportunities and options.
4. Framework approach has improved our flexibility and design.
5. Focus on communication mechanisms and data to be communicated help leverage industry trends and technologies.
6. We no longer simply try to pick best component in a problem domain; rather, users can evaluate and change selections as required.
7. Maintenance approaches are being reconsidered.
8. Building flexibility into a system leaves margin for growth and evolution.
9. It is important to have a marketing plan and the ability to explain the risks and benefits (good technology is not enough). What do you offer the customer?



Looked at from yet another perspective, GMSEC and other framework architecture approaches represent what may be a set of new trends in mission critical systems development. Some traditional approaches are on their way out, while other approaches are on their way in:

OUT	IN
Best Solution	User Choices
Identical solution	Common Solution
Component Repository, Reuse Library	Component Catalog
Socket Connections	Publish/Subscribe
Lots of Interface Control Documents	Message Specification Document
Custom interface software, COTS changes	Message adapters
Big development team	Small component teams
Daily integration meetings	Developers just integrate to the bus
Daily system recycles	Add/change components anytime
Integration intensive	Configuration intensive
Custom Applications	Configurable Applications
Reengineer whole system every 7 years	Replace/upgrade "stale" part each year
Independent status displays	Situational Awareness
Independent control points	System-wide control
Knowledge experts define automation	Ops team defines automation

GMSEC development and use is expected to continue for many more years. It is expected that the area of situational awareness and automation will continue to evolve, as GMSEC is only beginning to show the power "public" information on the bus. For use in larger networks and system-of-systems approaches, work will be needed in the areas of security and interoperability, as middleware products today do not consistently handle these aspects of large systems when middleware products are mixed.

Providing or giving away the architecture and API as open source may provide GSFC with one of the largest paybacks. This will expand the user base and therefore increase the number of compatible functional components which can be immediately integrated into GSFC operational systems. Some of these new components may be available for free and some may have a license cost, but the integration costs for GSFC would be near zero.

The real future directions for GMSEC are just now being determined. The end-users are now comfortable with the new levels of automation and are considering system enhancement needs. Other NASA Centers are considering using approaches like GMSEC and may want to help expand GMSEC into a NASA common approach. Other organizations (commercial and government) are considering the use of GMSEC and would help influence its direction. Many organizations would like to adapt their components to meet the GMSEC standards so they could be considered for GMSEC use. What is clear is that there are many options and the GMSEC architecture has the flexibility to expand in many different directions.

VII. Conclusion

The GMSEC Architecture features plug-and-play components, standard messages, and a software information bus. Components can be core functional applications such as Telemetry & Command, Planning & Scheduling, Assessment & Archive, Guidance Navigation & Control, and Simulation & Modeling or new stand-alone functions. In many cases, there are multiple components covering the same functional area so that a mission can select the component best matched to their requirements. The components publish/subscribe to the information bus using GMSEC standard messages. The GMSEC Application Programming Interface (API) shields the components from dependencies on communication protocols, operating systems, and hardware platforms thus facilitating platform transparency for the components. The API supports and "normalizes" the behavior of several communications middleware products, providing additional flexibility. Legacy components may interface to the Information Bus using adapters that translate legacy component messages and/or protocols to those that are GMSEC-compliant. Additionally, legacy components may, in combination with GMSEC, continue to use legacy interconnections.

The value of the GMSEC Architecture and tools is not just about dollar savings. Ground systems can now be built and prototyped faster, allowing for quick and low-cost "fly-offs" or evaluations of different ground system

components and implementations. New capabilities are enabled including more powerful and system wide automation. In the past, each domain had automation within their components, but there was no generic commonality or cross-domain sharing of information. The "public" nature of the information bus allows cross-domain knowledge of system status. In addition, upgrade approaches are more flexible inherently allowing parallel operations to verify before switching to a new version or component. Components can truly operate as black boxes with loose coupling simplifying any future change out of components.

The significance of the GMSEC architecture and supporting API middleware and tools can be viewed from a number of perspectives. For end-users, the architecture enables levels of automation beyond the current state-of-the practice on GSFC missions – thereby enabling missions to operate with smaller support teams. For system developers, GMSEC provides a way to quickly integrate large sets of functional components, allows for simplified technology infusion over time, and increases the ability to share software between missions. For COTS vendors, the ability to easily adapt their products using the GMSEC API allows them to demonstrate their products in a NASA environment (sometimes for the first time) and have an opportunity to be considered for NASA missions (several vendors have now made their first GSFC sales this way). For other NASA Centers and organizations (like APL and commercial satellite operators), GMSEC offers the opportunity for collaboration and tools sharing and provides a flexible framework for their consideration for their own systems. It is the combined technical and business aspects of GMSEC which has attracted the attention of so many organizations.

Acknowledgments

D. Smith, J. Bristow and J. Wilmot acknowledge the contributions of the entire GMSEC development team in developing the GMSEC technical solution. Credit also goes to Pat Crouse and Chris Wilkinson at GSFC, the primary funding sources and supporters of the GMSEC business model, and the management of the GSFC Information Systems Division for supporting the business reengineering efforts which have become a critical part of the GMSEC system success.

References

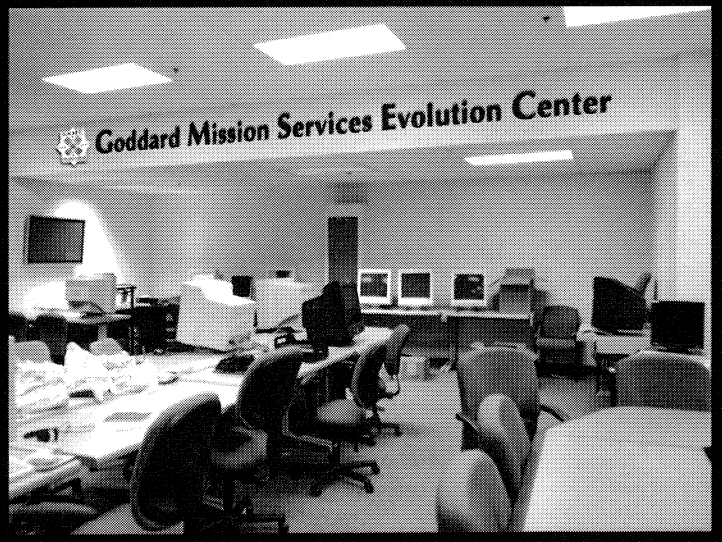
2006, IEEE Aerospace Conference, Lessons Learned from Engineering a Multi-Mission Satellite Operations Center Maureen Madden NASA/GSFC, Everett Cary Jr., Timothy Esposito Emergent Space Technologies, and Jeffrey Parker, David Bradley Honeywell Technology Solutions, Inc. 2006, IEEE Aerospace Conference

Bridging ESA and NASA Worlds: Lessons Learned from the Integration of hifly®/SCOS-2000 in NASA's GMSEC Jean-Pierre Chamoun, Steve Risner, Theresa Beech, Gonzalo Garcia , GMV Space Systems Inc 2006, IEEE Aerospace Conference

A Successful Component Architecture for Interoperable and Evolvable Ground Data Systems

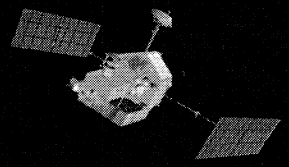
Co-Authors:
Dan Smith
John Bristow
Jonathan Wilmot

NASA Goddard Space Flight Center
Greenbelt, Maryland USA 20771



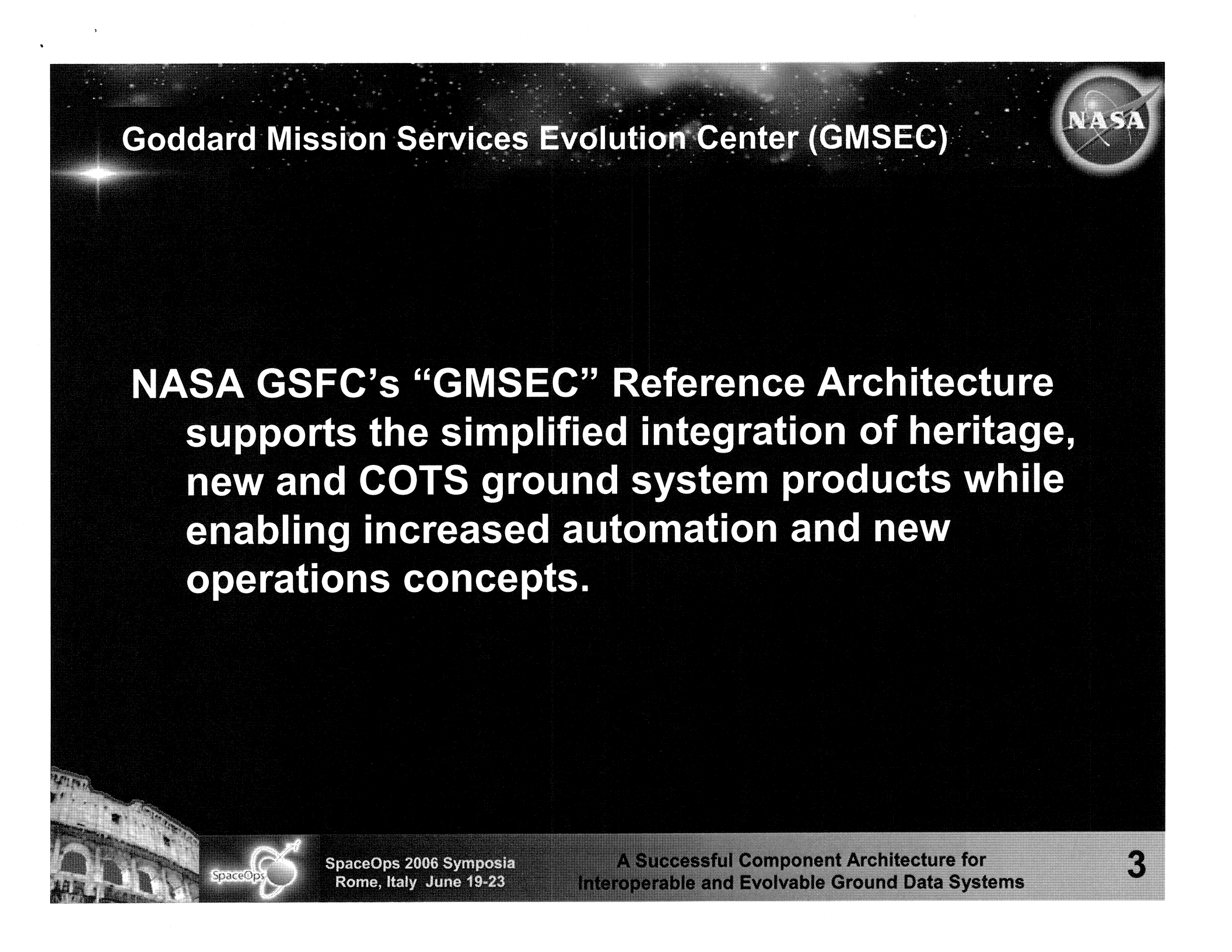
NASA/GSFC Mission Background

- NASA/GSFC manages about 30 spacecraft
  - ½ at the NASA campus in Greenbelt, MD
  - ½ at Universities around the United States



- Typical characteristics . . .
  - Scientific missions in low-earth orbit
  - Telemetry rates of <100Kbps, some higher for science recorder dumps (up to 150 Mbps)
  - Mission durations of 6 months to 20+ years
  - Each mission has its own control center and ops team
- Primary issues
  - Cost of development, ops and maintenance
  - Slow advancement of new capabilities and technologies
  - Very little commercial software (COTS)

Goddard Mission Services Evolution Center (GMSEC)

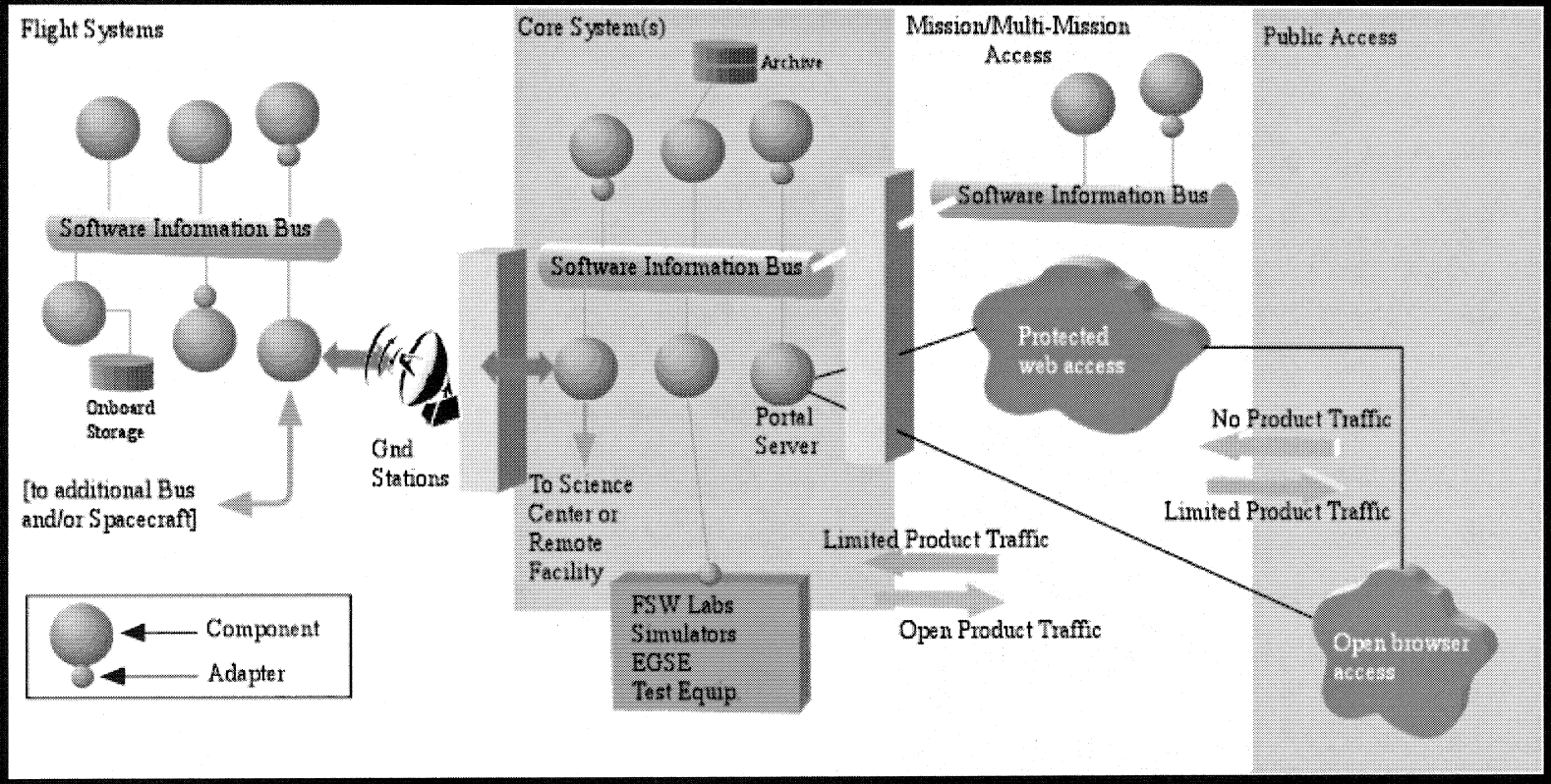


NASA GSFC's "GMSEC" Reference Architecture supports the simplified integration of heritage, new and COTS ground system products while enabling increased automation and new operations concepts.

GMSEC Approach

- Goals
  - Simplify integration and development
  - Facilitate technology infusion over time
  - Support evolving operational concepts
  - Allow for mix of heritage, COTS and new components
- Concepts
  - Standardize interfaces – not components
  - Provide a middleware infrastructure
  - Allow users to choose their products (no single answer)
  - General purpose approach with broad applicability

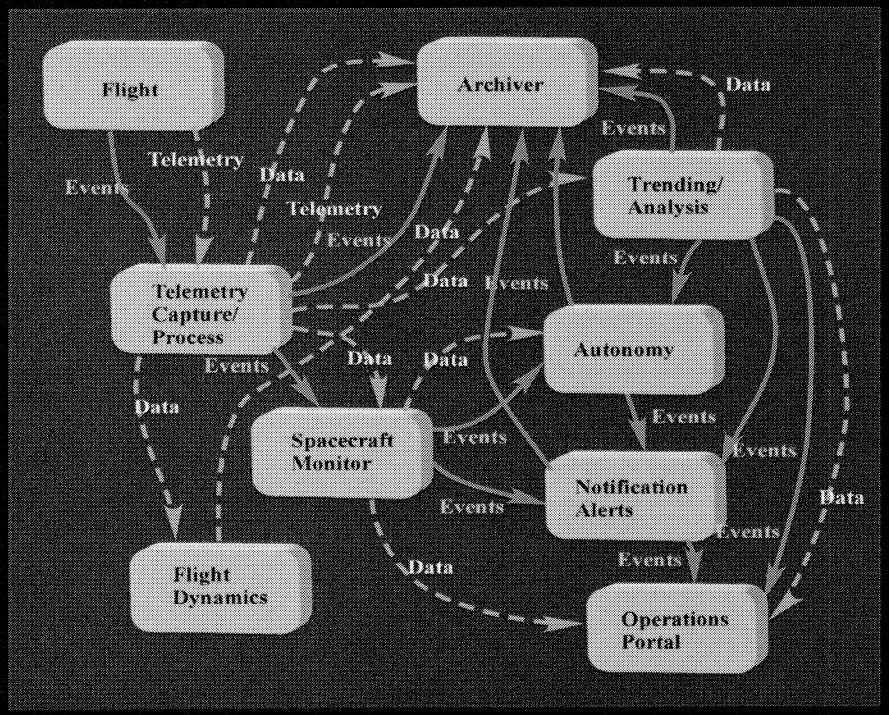
End-to-End GMSEC Vision



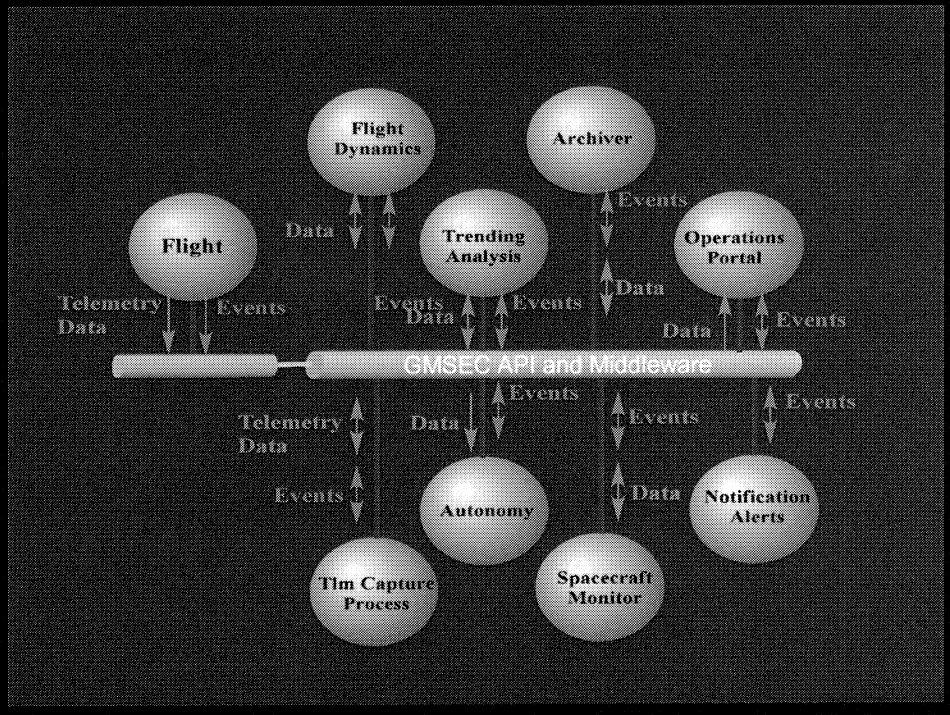
A middleware-based pub/sub framework extended from the space segment to the ground segment with extensions to provide web access.

GMSEC Publish Subscribe Communication

Traditional Design
Socket Connections



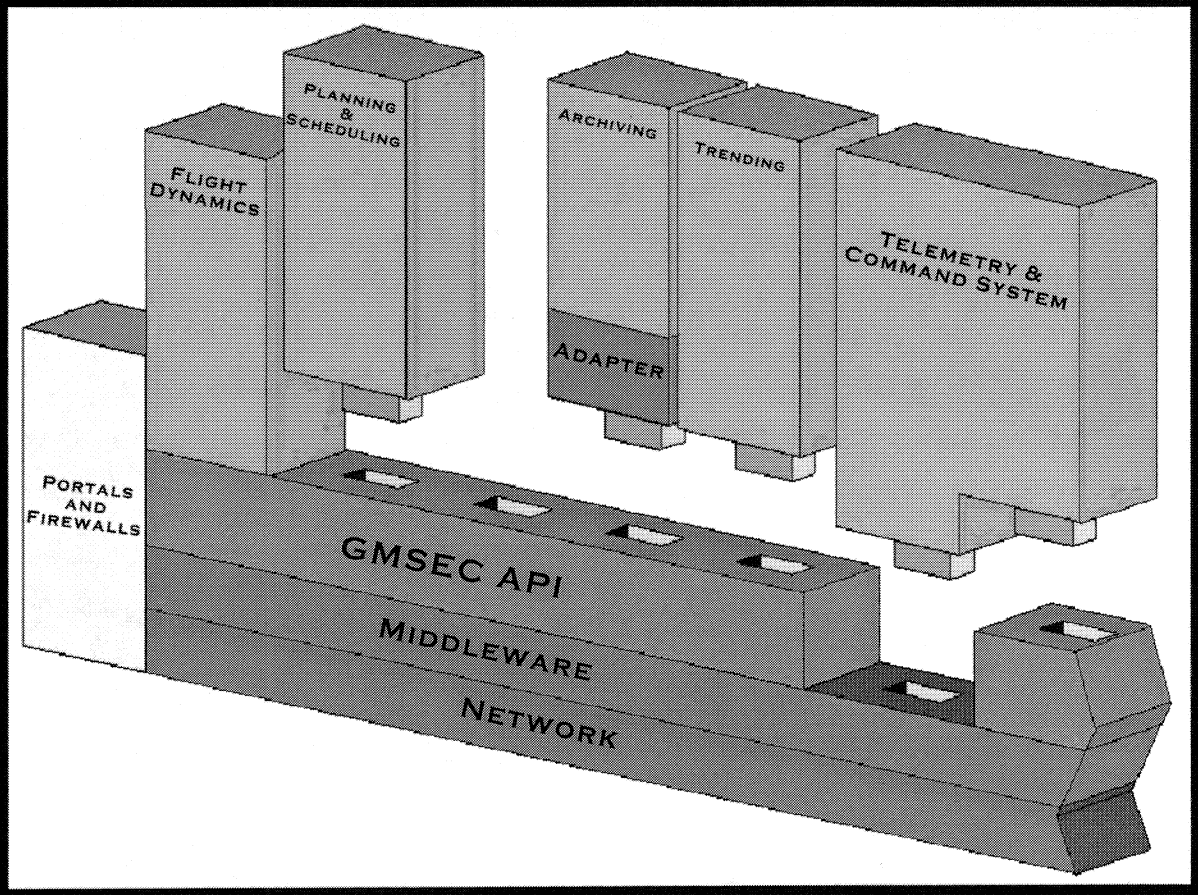
GMSEC Design
Middleware Connections



Middleware simplifies integration by having components interface to a bus and not to each other.

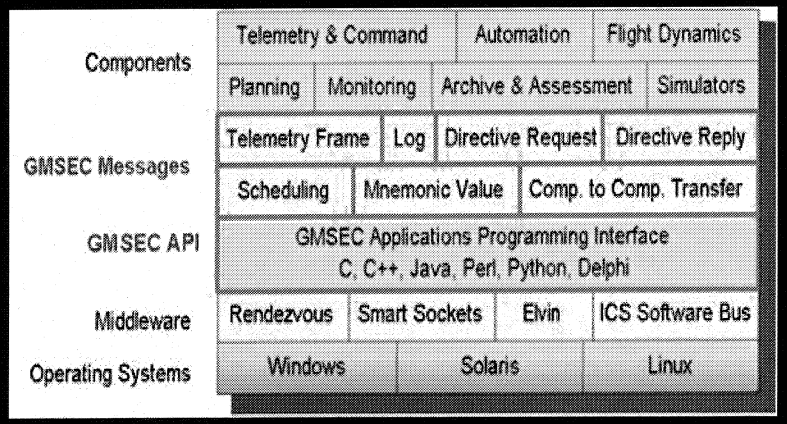
Plug-and-Play Concept

By creating a "framework", individual applications can be easily integrated into an existing system without regard to many underlying implementation details.

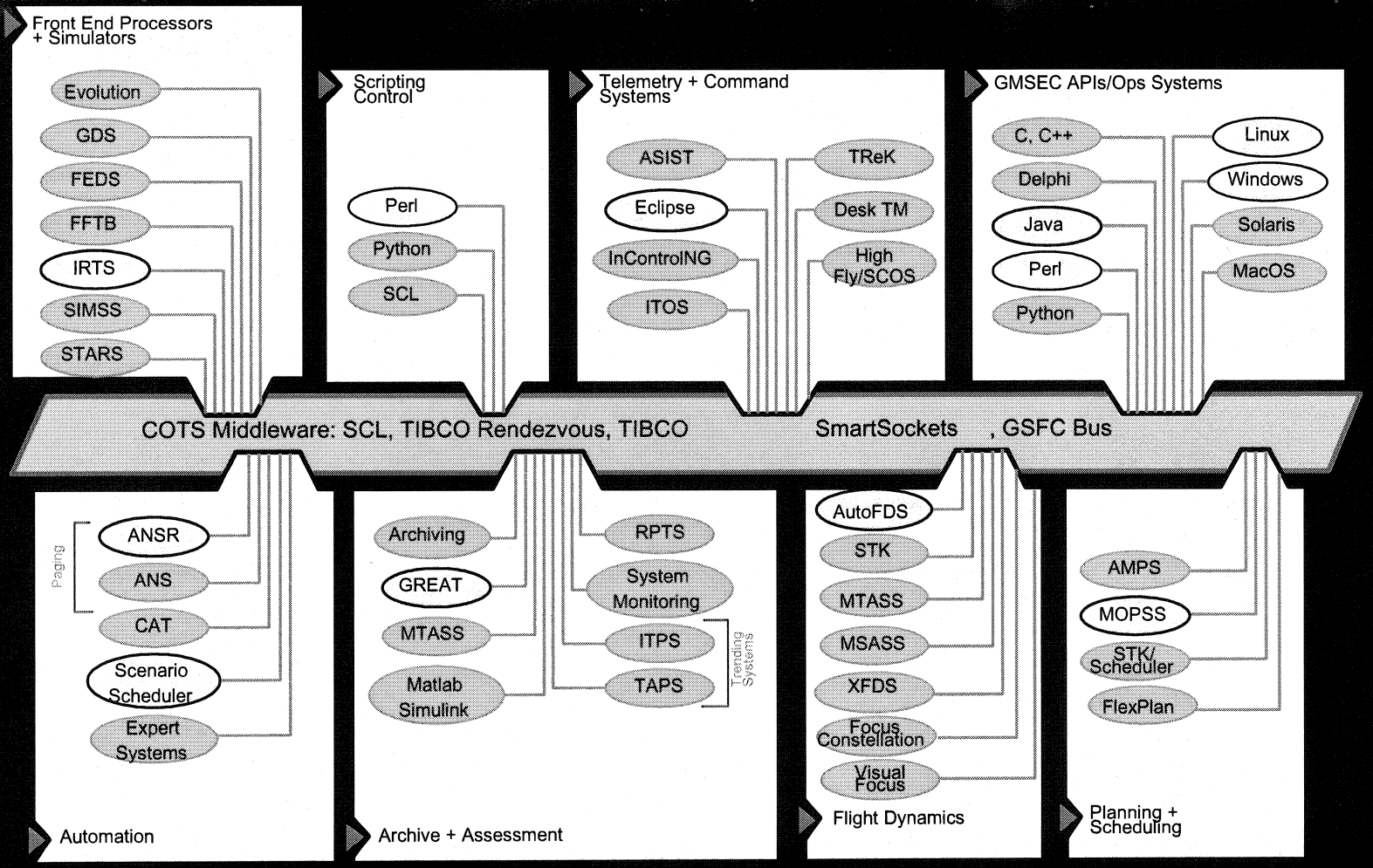


GMSEC System Technical Status

- Over 50 components available
- GMSEC Architecture Application Programming Interface (API)
  - Middleware Independent
    - Middleware options including: TIBCO SmartSockets, TIBCO Rendezvous, ICS SWB, GSFC Bus, IBM WebSphere, SOAP
  - Cross Platform API
    - Linux RH (7.1-7.3,9, + Enterprise), Suse (8.1)
    - Windows NT/2000/XP; Solaris (gcc + cc)
  - Multiple Languages: C, C++, Java, Perl, Python, Delphi
- Automated test package for over 12,000 combinations of middleware, languages, platforms, OS's
- Architecture, API and GSFC Bus approved for OPEN SOURCE release in April 2006



GMSEC Component Catalog



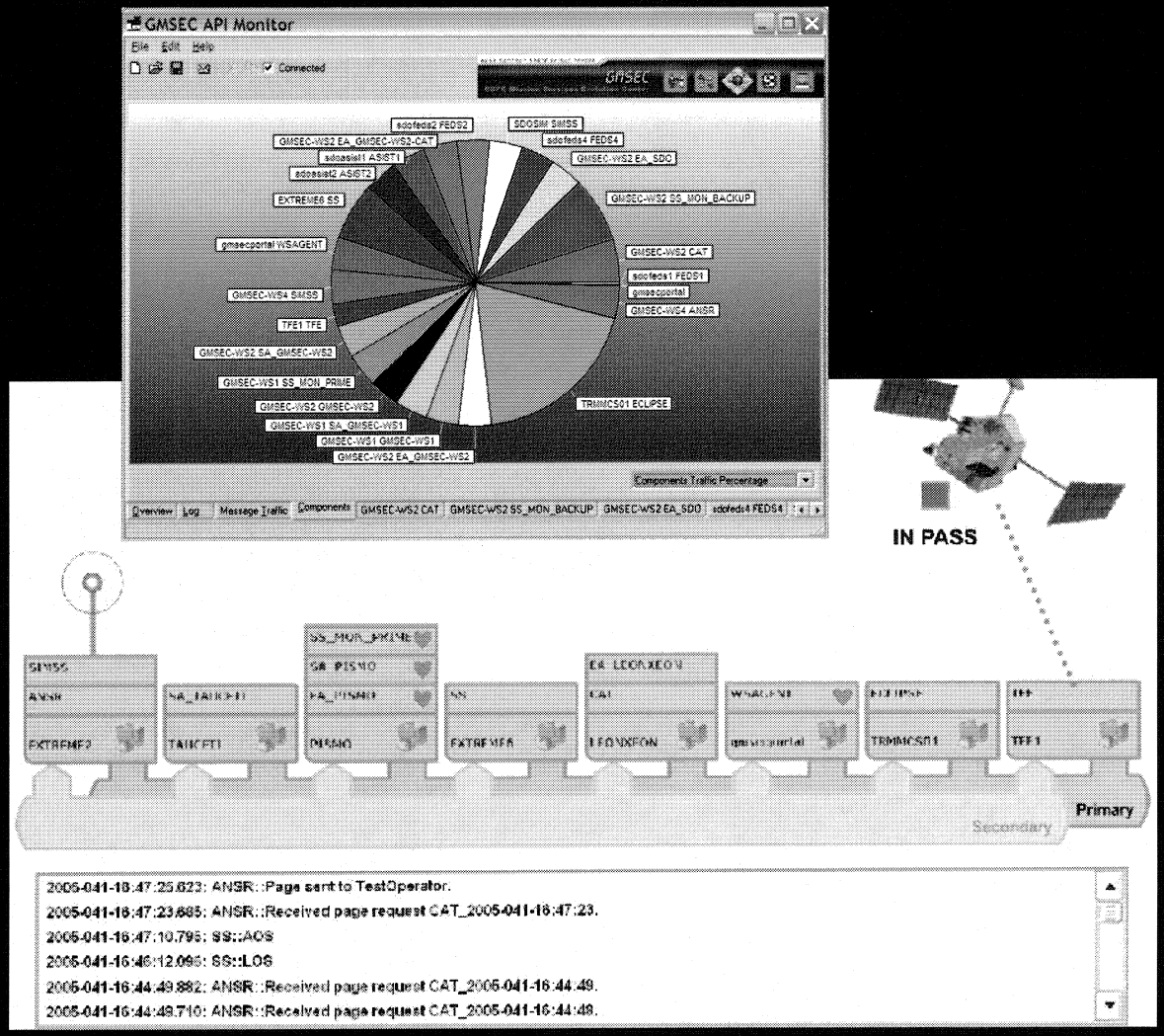
Choices are available for many subsystems. The TRMM mission selected catalog components to best meet their reengineering needs.

Automation Concepts

- Each Component Must Meet Certain Standards
  - Meet its functional interfaces
  - Publish keep-alive and status messages to the bus
  - Accept control directives over the bus
- Common Tools Cross Domain Boundaries
  - Tools can "listen" for status from all components
    - Provides system-wide **situational awareness**
  - Single tools can direct actions of any number of components
    - Provides **system-wide control** ability
  - "Criteria-Action Tool" provides ability to define situational awareness rules and corresponding actions
    - Allows for **event-driven automation**

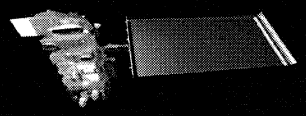
Tool Development is Simplified

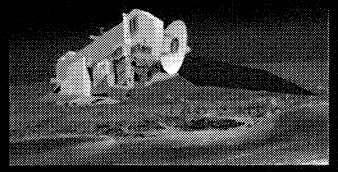
- Support tools are easy to develop
- May not require any integration with other components
- Simply monitor messages on the bus
- Examples
  - Performance tool
  - Configuration display
  - Message replay



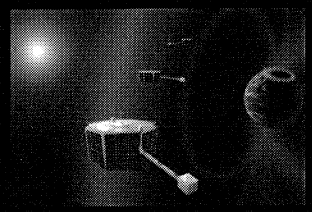
GMSEC Operational Status

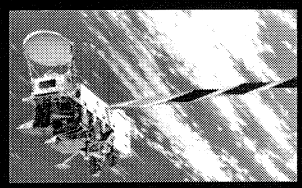
- Existing-mission reengineering efforts
  - Tropical Rainforest Measuring Mission (TRMM)
    - Goal: reduce operations cost by 50%
    - Operational with GMSEC architecture since Fall 2005
    - Pathfinder for Terra, Aqua, Aura automation (2006-2007)
  - Small Explorer (SMEX) missions – SWAS, WIRE, TRACE, SAMPEX
    - Operational on SWAS, preparing for others
    - Conducted a successful 2-week lights-out operation
    - Pathfinder for low-cost fleet operations & updating existing space science missions
  - ST5 3-Satellite Constellation system - Launched March 2006
    - Technology demonstration with subsystem modeling and closed-loop automation





- **New GSFC missions**
  - Working with 6 future missions





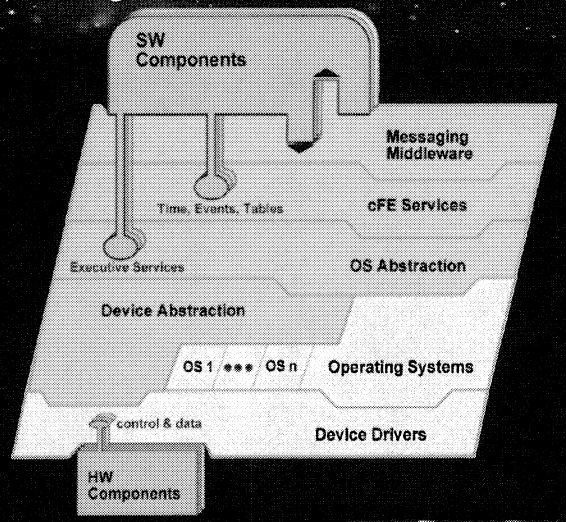
- **Other applications**
  - GMSEC Labs installed at other NASA Centers, involved with Exploration Initiative
  - Commercialization/Other interest: Other U.S. government organizations, several major commercial satellite operators, some interest beyond the space domain

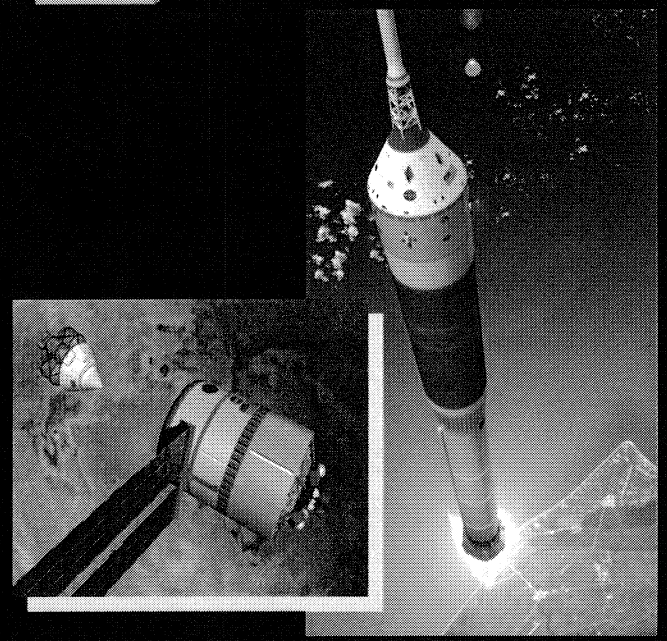
Lessons Learned / GMSEC Benefits

1. Significant reduction in integration time
2. Components added/upgraded without impacting existing system
3. Ideal for using multiple small distributed development teams and vendors
4. New concepts emerging for small independent components that integrate with the bus and provide immediate benefits
5. Missions more willing to adopt the approach if "old favorite" components can still be used
6. Some vendors see message compliance as a way to enter what had appeared to be a closed marketplace
7. Standard message approach provides collaboration possibilities with other organizations
8. We have become smarter shoppers for products with clean interfaces and modular functionality
9. The architecture itself promotes advanced automation and ops concepts
10. The same concepts can apply to ground, flight, or other domains

Future Directions

- Increased security and interoperability provisions
- Continued coordination with Exploration Initiative
- Continued extensions to flight and remote access
- Evolution of technical capabilities
  - Situational awareness
  - Automation/autonomy
  - Data mining
  - Network/system performance tools
- Resolve implications of emerging standards
- Expanded domain application





Conclusions

- Message-bus component-based framework architectures are well proven and provide significant benefits over traditional flight and ground data system designs.

- Missions benefit through increased set of product options, enabled automation, lower cost and new mission-enabling operations concept options.