SpaceOps 2010 Abstract Form

Do not extend beyond this one page.

Do not change font size (11).

Text-based symbols are OK, but embed fonts

*Graphics are **not** OK.*

Read the Author's Kit for more details.

Tell us your presentation preferences:

Add only **Y** for Yes in the brackets **[ ]**, N's are not needed. We encourage flexibility - both oral and poster forums have their strengths. See website.

- I can present in either oral or poster sessions **[ ]**
- I will only present in an Oral Session **[Y]**
- I will only present in a Poster Session **[ ]**
- I would like an ePoster Session because my topic suits that forum **[ ]**
- If selected as a poster presenter, I will consider a request to switch to an oral presentation to cover for a withdrawn oral presenter **[ ]**

Keywords: (add keywords that describe your topic)
Architecture, data systems, message standards

Your Abstract Title: (Should be the same as your online submission and your future manuscript title – 12 word limit)

Message Bus Architectures – Simplicity in the Right Places

Your Author list: (each author's name and affiliation)

Dan Smith, NASA Goddard Space Flight Center, Greenbelt, Maryland

Your Abstract text:

NOTE: This presentation is intended to complement, and draw contrasts to, a paper being submitted by Nestor Peccia from ESA on the advantages of a Service Oriented Architecture. It is hoped that the two papers may be presented as a pair with a combined question/answer period at the end.

There will always be a new "latest and greatest" architecture for satellite ground systems. This paper discusses the use of a proven message-oriented-middleware (MOM) architecture and the strengths it brings to these mission critical systems. Service Oriented Architectures (SOA) are part of the latest trend in advanced system design and are generally considered more powerful than the MOM approach. A MOM vs SOA discussion can highlight capabilities supported or enabled by the underlying architecture and can identify benefits of MOMs and SOAs when applied to differing sets of mission requirements or evaluation criteria.

NASA's Goddard Space Flight Center (GSFC) has been developing a message oriented architecture since 2001 and has been operating satellites with the system since 2005. The Goddard Mission Services Evolution Center (GMSEC) architecture is now being applied outside of NASA/GSFC and is being used for systems other than satellite control. The GMSEC concept involves a mature Applications Programming Interface (API), a messaging middleware, and a set of defined message specifications. Instead of building GMSEC software components to meet detailed satellite control functional requirements, the GMSEC concept is to encourage component providers and innovators to adapt their existing products to match the GMSEC interfaces. This open approach has resulted in dozens of commercial products being made "GMSEC compatible". These "plug and play" components cover a wide range of functionality from most of the control center product vendors. In fact, the GMSEC effort has not involved development of any core software applications for telemetry decommutation, command generation, or data trending – these functions are met by existing applications.

Key areas of interest for evaluating system capabilities will vary by user. Some of the important considerations for the GMSEC approach relate to the following:

a. Time to deploy

b. Selection of available products

c. Role of the COTS providers

d. Ability to work with legacy systems

e. Technology infusion over time

f. Handling of enterprise and cyber security requirements

g. Use of Standards

Although the status of the GMSEC system implementations will be presented, the primary purpose of the paper is to highlight the benefits and enabling capabilities of the architectural approach and where the MOM and SOA approaches may each have their place in advanced satellite ground system developments.

[Submittal of final paper contingent on NASA review and approval.]

Message Bus Architectures – Simplicity in the Right Places

Dan Smith[1]
National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland, USA

There will always be a new "latest and greatest" architecture for satellite ground systems. This paper discusses the use of a proven message-oriented middleware (MOM) architecture using publish/subscribe functions and the strengths it brings to these mission critical systems. An even newer approach gaining popularity is Service Oriented Architectures (SOAs). SOAs are generally considered more powerful than the MOM approach and address many mission-critical system challenges. A MOM vs SOA discussion can highlight capabilities supported or enabled by the underlying architecture and can identify benefits of MOMs and SOAs when applied to differing sets of mission requirements or evaluation criteria.

Nomenclature

API = Application Programming Interface
CCSDS = Consultative Committee for Space Data Systems
COTS = Commercial Off-the-Shelf
GMSEC = Goddard Mission services Evolution Center
GSFC = Goddard Space Flight Center
MOC = Mission Operations Center
MOM = Message Oriented Middleware
NASA = National Aeronautics and Space Administration
SOA = Service Oriented Architecture

I. Introduction

It seems that new approaches for software system architecting and development are popularized every couple years and the pace seems to be increasing. We've come a long way since all of the satellite control center software was written in assembly language and entered on punch cards. We've used FORTRAN, Ada, C/C++, and JAVA. We've practiced Structured Design and Analysis and Object Oriented Programming. We've tried different approaches to networking, socket connections, software reuse, use of commercial products, and product lines. Today we talk about frameworks, virtualization, and clouds.

Many of the new trends of the past several years deal with frameworks. With a framework approach, a software system is developed to support many of the core functions of a system or software component. Functional components can then be more simply developed, integrated, and reused. Message-Oriented Middleware (MOM) systems and Service Oriented Architectures (SOAs) are two framework approaches that can be compared and contrasted in the context of mission-critical operations centers for satellite control.

A Message-Oriented Middleware reference architecture is in use at NASA's Goddard Space Flight Center (GSFC) and has proven very successful where it is applied. Part of its success is based on how it matches up to the

[1] GMSEC Project Manager, Software Engineering Division, MS580, 8800 Greenbelt Rd, Greenbelt, MD 20771 USA

specific challenges at GSFC. After several years of use, it is clear that MOMs present a highly beneficial approach to meeting GSFC mission operations center requirements, but also that it is not the best approach for all applications. This paper describes the GSFC MOM and its benefits, and looks at SOAs as a potential alternative or supplemental approach.

A. The NASA Goddard Space Flight Center Environment

NASA GSFC [Fig. 1] manages about 30 scientific satellites at any time, mostly in low-earth orbit. One half of these are typically controlled from the GSFC facilities in Greenbelt, Maryland, USA and the others are at various Universities across the United States. On-orbit life varies from a couple of months to ten years or more.

The GSFC satellite missions are typically individually funded and have their own dedicated control centers. In only a few cases are multiple satellites managed by a shared team, although the long-term plan is to move towards multi-mission control centers.

The large number of satellite missions at GSFC provides a rapid technology cycle for trying to advance new ideas, but also has led to a large collection of different products collected over a period of many years through the development of many mission control centers. Although much of the software is often reused, there is no single software suite used by all of the missions. Commercial Off-the-Shelf products from multiple vendors are also being used to differing degrees on different missions.

***Figure 1**. NASA's Goddard Space Flight Center controls many of NASA's unmanned scientific satellites.*

B. SOA Concepts

Service Oriented Architecture frameworks [Fig. 2] emphasize the use of functional services to simplify the development and integration of software components.

There is no single definition of Service Oriented Architecture. In general terms within this paper, SOA refers to an architecture in which functional capabilities are implemented as services available to other components within the system or made available for external access. Through well defined interface agreements, service provider components can implement a capability at the request of a service consumer and provide actions or data in response to the request. Common needs of functional components, such as security checks, access to the current time, and data archiving may be implemented as core services that are utilized by many of the larger services in the system.

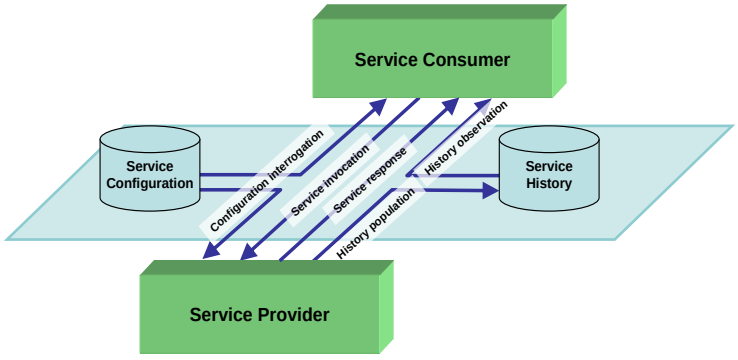

***Figure 2. Service Oriented Architecture.** The SOA model as defined in the CCSDS Spacecraft Monitor and Control documentation[1].*

The SOA model is very powerful in its ability to integrate many components into a common environment with consistent policy guidelines for each component in the system. A service registry is used to identify available services, service details, and their locations. Through use of the registry, service components can be easily distributed or even moved and new services can join the system at any time.

C. MOM Concepts

Message-oriented middleware frameworks [Fig. 3] emphasize the data messages themselves and the power of publish/subscribe message distribution. Whereas in service-oriented systems service providers and service consumers are coupled through defined interfaces and functional operations calls, in a MOM the applications interact with the messaging system.

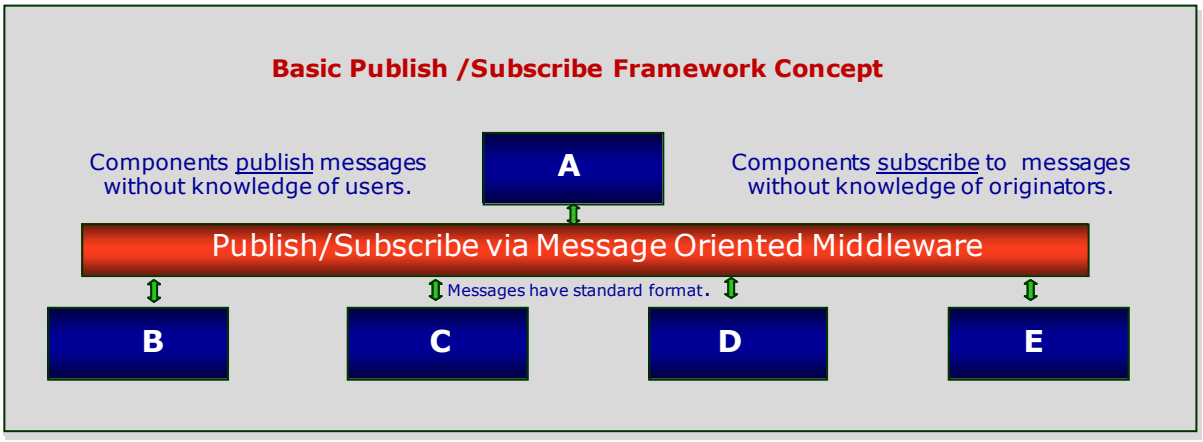

Figure 3. Pub/Sub Framework. *With publish/subscribe frameworks, applications integrate with the message bus. This approach is applicable to many system applications.*

With a publish/subscribe system, an application posts messages to the message bus without knowledge of which applications require it. An application may publish a log message indicating that a certain problem occurred at a specific time. The publishing application does not need to know that there are many software components that have subscribed to log messages. One may ask for log messages to drive text display pages, one to create a message archive, and one to raise an alarm or e-mail someone with an indication of the problem. Still another application may have subscribed to all of the log messages to drive an automated response system to try and resolve the issue. New applications can be added that subscribe to the same messages without any new configuration or interaction with the publishing component.

Similarly, if an application is interested in alarm messages "from anywhere in the system", a single subscription to log messages on the bus is all that is needed. There is no need to place service requests with each application with the potential of reporting an alarm message.

As many applications are added to the software bus, the power of the publish/subscribe approach increases further. If a new application is added which may generate alarm messages, no changes and no new service calls are needed in those applications monitoring all of the system alarms.

With the MOM, other interaction patterns, including Prompt-Response or Point-to-Point, can also be implemented.

II. The NASA "GMSEC" MOM Architecture

A. Background

Several challenges emerged in the 1990's from GSFC's approach of having each mission responsible for development of its own mission operations center. Innovation was slowed as each budget-constrained mission worked to meet only its own requirements and to minimize additional system enhancements. Studies, designs and implementations efforts were often repeated for each new mission. COTS product lines were typically not even considered because they had never been integrated with other GSFC products.

Common approaches to large system development were adding to the problems. Each software component required direct communications paths with each other component with which it shared data. Although this "socket connection" design solved some of the system-wide miseries of the earlier "shared common" approach, it made it very difficult to add, replace, or modify components as the system grew and became more complex. At the time, the concept of frameworks and middlewares were gaining acceptance in other industries [Fig. 4], but were not generally accepted for satellite control centers.

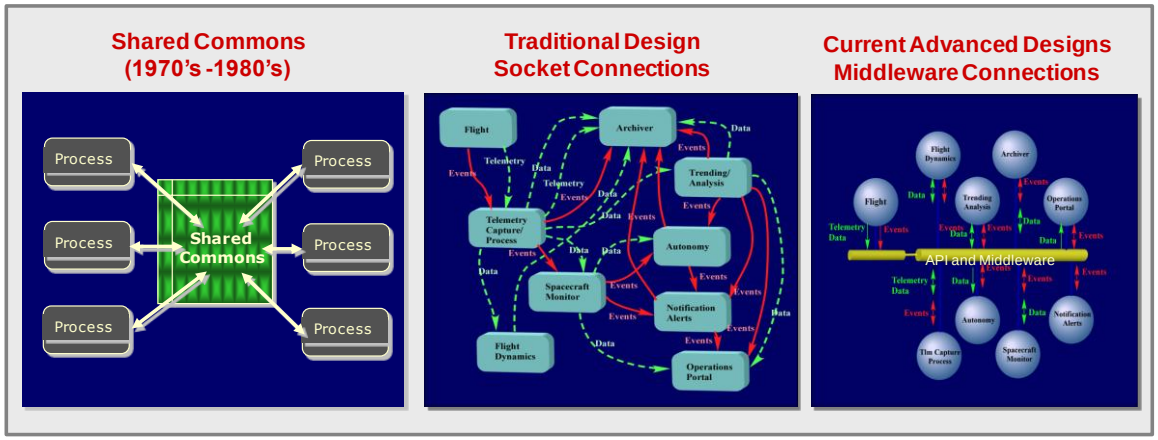

Figure 4. Advances in Development Approaches. *Framework architectures provide benefits over previous approaches to satellite mission control center development.*

The problems with the traditional approach were made worse with the increased cost pressures of the late 1990's and the outlook for a healthy set of planned missions. The decision was made in 2001 to begin development on reference architecture for future missions. The new paradigm would represent a significant change from the previous approach of integrating selected components to create mission-unique systems. The new architecture, named the Goddard Mission Services Evolution Center (GMSEC) ground system, became operational in 2005 and has supported many missions at GSFC since that time[2].

GMSEC is not trying to select the "best of breed" component in each functional area. The GMSEC team is not trying to compare COTS products against each other or against a heritage system. Instead, the architecture allows the user to select the most appropriate products based on functional need or personal preference and easily integrate them into a ground system. Keeping this responsibility with the mission teams has greatly increased the acceptability of the GMSEC approach.

The GMSEC framework has also been applied recently in other areas, including the NASA GSFC Flight Dynamics Facility, and is a core part of evaluation labs at locations outside of NASA GSFC.

D. Driving Goals of the Framework

Goals of the new GMSEC framework were discussed and documented prior to selection of an underlying technology or design. The primary goals addressed the key challenges recognized in how mission control centers at GSFC were designed at the time and the need to accommodate many new missions and new operations approaches in the near future. The goal in 2001 was to develop a system which could be used for mission development efforts for up to the next 10 years. Four key goals were identified:

1. **Reduce system deployment time for new missions.** The framework should allow for simplified integration of the many software components utilized in a mission operations center. It was further decided that the simplified integration goal would focus on large component granularity – such as a full telemetry and command system, planning system, trending system, etc. The goal was not to focus on widget-level, or small routine level integration; although it could be a side-benefit.

2. **Allow for the increased use of COTS and GOTS components.** GSFC could no longer afford to build all of its software from scratch when multiple commercial products were available. Often, the COTS products were advancing at a faster rate than the similar in-house products. However, the issue of vendor lock-in was a major concern. Products from different vendors would need to be used in the same system and it should be assumed that one product may need to be replaced for another sometime during the mission life. A key metric derived from goal #1, above, would be the time needed for a COTS product to be integrated into a GMSEC-based system.

3. **Allow for the infusion of new components over time.** This goal addressed two key challenges facing GSFC missions. First, it was very hard to add new capabilities to well-established systems. With tightly-coupled software components, it was hard to make significant changes are integrate a major new component. The easiest approach, therefore, was to work with the original system functionality and make only the most critical changes over the life of the system. Secondly, when a new software component or significant update was implemented for a given mission, it was very difficult to then apply that same change to any of the other mission systems.

4. **Enable new capabilities and operations concepts**. The framework should provide a growth path to increased levels of automation and the ability to combine status information from multiple components (now termed "situational awareness"). In addition, GSFC was anticipating the need for new operations concepts, including multi-mission control centers, constellations of satellites, distributed operations (possibly split between the GSFC campus and a university), etc.

E. GSFC MOM Architecture Concepts

Key technical aspects of the GMSEC architecture are the publish/subscribe message bus, the Applications Programming Interface, the use of standard message formats, and the ability to integrate both existing functional software components and new components with capabilities enabled by the architecture itself [Fig. 5]. Each of system attributes are described below.

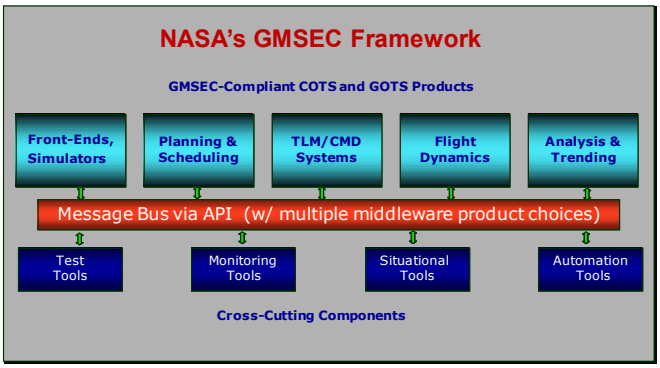

Figure 5. GMSEC Framework. *Multiple middleware choices are accessed through the GMSEC API.*

1. The Message Bus and API

The GMSEC Architecture uses a middleware message bus (sometimes called an

information bus or software bus) for inter-process and inter-node communication. The middleware keeps track of where processes are located and which process requires the data when it is 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 normal 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.

Although the publish/subscribe message patterns 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 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 a change to the applications software and vendor lock-in is avoided for the underlying middleware. This middleware flexibility allows for product swapping if necessary, but also allows for low-cost middleware to be used for development, high-reliability and high-performance middleware to be used for operations and small-footprint middleware to be used for flight – all with the same functional behavior.

The GMSEC API provides isolation between the applications programs and the underlying messaging software. As discussed above, any of several different middleware packages can be used without modifying the applications. In addition, the API supports multiple languages, operating systems and platforms [Fig. 6]. It normalizes the behavior of the middleware while allowing access to special functions or capabilities of individual middleware products.

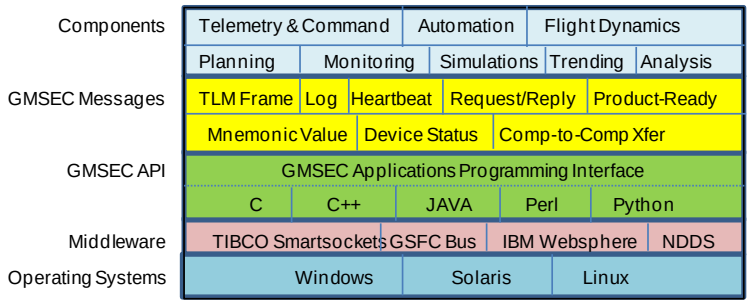

Figure 6. Layered Architecture. *Multiple operating systems, middleware choices, and languages increase the flexibility of the framework.*

2. Standard Message Formats

GMSEC standard messages meet the needs of the key interfaces for mission control applications. Additional messages may be created as needed. A subject name is specified with each message published via the API. The subject is used for routing to the subscribing applications and contains information including the mission identifiers, nodes, message type (e.g., telemetry), etc. The messages themselves include a common header used by all messages followed by a message-specific body. The message header contains some of the same subject information, but also includes time stamps and more details on the message type. The body of the message may contain a telemetry frame or packet, a text message, user directive, archive request, etc. Data from multiple missions can be on the bus at the same time and distinguished by different values in the header fields[3].

All of the message formats are defined in the GMSEC Message Specification Document. Vendors match their product interfaces to those in the document and make calls to the API to receive or send messages. Individual interface control documents between components are not needed, although the behavior of coordinating processes should be documented.

3. Compliant Components

The use of common interface message formats allows many different products of the same functional domain area to be integrated. By having choices in each functional area, missions can avoid vendor lock-in and can select each component based on its own merits (functional, technical, cost, etc.). Components can be as major as telemetry and command systems or planning systems or as small as performance monitoring tools.

An "adapter" approach is used for the existing in-house or commercial components. The adapter is a piece of software which works like an API-to-API interface and converts from the existing package's interfaces to the GMSEC interfaces. Because the GMSEC interfaces were developed with knowledge of many COTS interface definitions, this adaptation has been proven to go very quickly (from a day to about 2 weeks).

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

- It must meet its functional requirements; although verification responsibility is with the missions
- It should publish a heartbeat message on a periodic basis
- It should publish status/log messages to indicate an action has taken place or an event has occurred
- It should allow user directives for the component's control to be received over the message bus

These simple rules can yield very powerful results and enable new types of cross-cutting functional components. The heartbeats allow for system monitoring, configuration displays and automated failovers. The log messages across multiple tools provide a new level of situational awareness. Allowing directives to be sent to any component allows for scripting and, combined with situational awareness, provides new levels of reactive system automation.

F. **Status, Benefits and Weaknesses**

The GMSEC framework has been used for both new and reengineered mission operations control centers at NASA GSFC since 2005. The first three missions each used a different telemetry and command system product, but they used the same automation tools and alert system. This flexibility proved that a core goal of the effort had been met. In addition, there has been cost reduction in the operations effort as automation has allowed for the elimination of off-shift support. Observed benefits have included the following:

1. Automation for cost and risk reduction is the #1 selling point
2. Most commercial command and control products are now GMSEC compatible – increasing choices for the missions
3. Significant reduction in integration time
4. Components added/upgraded without impacting existing system; can support parallel testing
5. Ideal for using multiple small distributed development teams/vendors
6. New concepts emerging for small independent components that integrate with the bus and provide immediate benefits
7. Standard message approach provides collaboration possibilities with other organizations
8. Enables new approach for maintenance of very long-term systems
9. Basic framework is applicable for systems other than satellite mission control

Although the initial goals for GMSEC have been met and the benefits proven to be far reaching, there are some capabilities now being requested or considered for which the message approach does not provide a simple solution. A common approach to security (authentication and encryption) is a challenge when mixing products and maintaining the desired flexibility and integration simplicity. Also, GMSEC has been tailored to the needs of the independent control center environment. Enterprise communications, where controlled data sets are shared with selected other external systems needs to be done without the wide-open and higher-bandwidth simplicity of the publish/subscribe paradigm. All of these limitations can be overcome, but it may require making the message-oriented approach look more like a point-to-point or service oriented system in places.

III. What is the Best Framework Approach?

There have been very successful mission control center development efforts for decades. Clearly, there is no single way to architect these mission critical systems. Given multiple powerful choices today, which one should be selected? The answer probably depends on a number of key factors, and there still may not be a single "best" answer.

A. It may depend on your organizational characteristics and goals

Although the GMSEC design team selected 4 key goals, the goals had to be addressed within the context of the NASA GSFC requirements, culture and organization:

- **Risk Averse**. NASA, although technologically advanced, is also very risk averse. Any new approach must be fully vetted and proven, at least in other mission critical environments, prior to applications to satellite control. The "newest and greatest" possible approaches are often avoided for this reason.

- **Data driven**. Much of the work of satellite control systems is triggered by the receipt of data – often satellite telemetry or system alarm messages. MOMs are ideas for dealing with data-driven systems. The concept of putting data streams from multiple satellites on the message bus and having the user software select which streams to monitor was part of the GMSEC plan.

- **Very short to very long duration missions.** NASA GSFC flies missions that may only last a couple of months, others may last ten years or more. The system architecture must be resilient enough to adapt to changes over a long period of time and yet simple enough to accommodate the very short-term low-cost missions.

- **The framework is not for a single system**. NASA manages dozens of space missions. Any investment in a framework must be able to accommodate many different uses over a long period of time. This is one reason that GMSEC allows for the swapping out of middleware systems and the choices for functional components.

- **Not from Scratch**. If the software is designed and built from scratch, one has total control over the partitioning of the system and the creation of functional services. For NASA, the goal was to take advantage of existing products with minimal effort. In many cases, products can not be easily decomposed to their service level granularity.

- **Not ready for "one size fits all".** Some organizations have realized great benefits from using common software for many missions. The range of missions and the culture at NASA GSFC make this difficult. The framework design had to accommodate a wide range of possible applications.

Additionally, there are other considerations or goals that may be important when considering which architecture to implement:

- **Availability of trained workforce.** Some approaches are so new or so unique that finding or training a staff for long-term support may not be practical.

- **Distributed nature of the environment.** Different approaches may be more conducive to distributed processing and communications. SOAs are generally considered better for remote service-based applications. MOMs work very well for some cases of splitting operations between two sites.

- **Selection of available products.** Simple approaches can be taken where specific products support functional areas. If, however, there will be many choices for a given function, a level of flexibility is

needed. Very specific service definitions may not be sufficiently flexible to accommodate the beneficial features of a wide variety of similar products.

- **Handling of enterprise and cyber security requirements**. Special effort may be needed to address system interoperability and security where missions involve partnerships, remote interaction, or enterprise-level communications. Data filtering flow control, authentication, encryption and bit-level message agreements may all be required.

- **Standards, policies, and common practices**. By adhering to standards, development efforts can be simplified, systems made more flexible, and a broader set of support or applications software may be available. Standards may be at the lowest communications level, may be at the message format level, or may even involve the definition of specific services or capability sets. Similarly in-house policies may limit certain communications or security choices. Following common practices, although not defined as formal standards, often simplifies the total effort and can increase the acceptance level of the design choices.

B. It may be that many approaches can work

Although the MOM approach has been successful at NASA GSFC, it is not the only approach used. Most of the older missions at GSFC utilize a pre-framework approach involving socket connections between components. Although most components were developed at GSFC, there is some use of COTS products. In general, this approach has worked well, partially due to the small number of key interfaces. The approach, however, is recognized at being week in areas that the MOM approach is strong, including maximizing reuse, situational awareness, automation, and multi-mission support.

Web services have been discussed extensively for use within GSFC. A primary data system could process telemetry and post data to a server where it could be accessed in a number of ways via web services. Cultural and security issues have prevented the widespread use of this approach for remote access to the data. This approach, however, has merits within the boundaries of the control center, without ever being exposed to the web. Still, it is not in common use at GSFC and would require redesigning major components for which there are no driving concerns.

C. It may be that one approach is best only in some areas

SOAs probably are discussed most often as a solid framework architecture to consider. SOAs are ideal for packaging major functions such as flight dynamics or remote data access and smaller services can be defined for common functions such as time and archiving. The use of a dynamic registry allows for the introduction of new services or the redistribution of existing ones. Although powerful in some applications (i.e. new applications showing up for stock market analysis), most control center systems are tightly controlled and the registry flexibility is not needed or desired.

Hybrid systems, with MOM, SOA, and Socket approaches selectively used to their best advantage may be the best solution as the capabilities of satellite control centers are advanced and expanded. MOMs are idea for localized component integration, situational awareness, and event-driven automation. SOAs are great for hiding the implementation of remote functional systems, and socket connections work well for passing data within individual components. For interconnecting remote systems to share data, direct network connections and file transfers between systems may continue to provide the most control.

IV. Evaluation of NASA's Framework Selection

NASA's GMSEC architecture is very well suited to the requirements, environment, and culture of NASA GSFC.

The message oriented middleware is ideally suited to data-driven systems. The emphasis on COTS product choices and standardized interface messages has led to a system which simplifies integration and allows for the user to select from many different products in the same functional area. With all components reporting status and publishing values on the message bus, new situational awareness tools and automation approaches have been developed.

For NASA GSFC, the ability to products from many different vendors and from the broad set of GSFC-developed software has been very beneficial. It reinforces the idea that different missions may have different needs and also that different operations teams may have differing preferences. To some, this is also a drawback. The concept of common solutions and operations approaches as a way to lower cost and increase consistency across missions and teams is also compelling.

At the point in time that the GMSEC architecture was developed, it probably was "a best solution". Although planned for ten years of useful life, the message oriented approach shows no sign of needing replacement. Instead, the potential of the approach is still being developed. At the same time, however, analysis is continuing on how to best address the new challenges brought on by enterprise data sharing and cyber security requirements. It may be that an architecture based solely on a MOM architecture will not address all of the new requirements being placed on the systems.

V. Conclusion

For where it has been applied at NASA GSFC, a message-oriented middleware framework for mission control center development has proven very successful. As a relatively simple concept, publish/subscribe messaging is ideally suited to the data-driven nature of satellite control and the easy integration of existing or commercial products. At the same time, it is also clear that successful systems can be developed around SOA architectures and that the service concept may even add value to the MOM approach, helping to meet requirements involving remote access and enterprise data exchange. However, both MOM and SOA approaches may be replaced by whatever "new and better" methodology or approach is next to gain popularity. Already, clouds and virtualization appear to be ready to change how we think about system design and integration.

References

[1]*Mission Operations Service Framework—Reference Model*. Draft Recommendation for Space Data System Standards, CCSDS 000.0-R-0. Red Book. Issue 1. Washington, D.C.: CCSDS, April 2009.

[2]Smith, D., Bristow, J., Wilmot, J., "A Successful Component Architecture for Interoperable and Evolvable Ground Data Systems", *AIAA Space Ops 2006 Conference*, Rome, Italy.

[3]Madden, M., Cary, E. Jr., Esposito, T., Parker, J., Bradley, D., "Lessons Learned from Engineering a Multi-Mission Satellite Operations Center*", IEEE Aerospace Conference*, Big Sky Montana, 2006.

[4]Smith, D., Grubb, T., Esper J., "Linking and Combining Distributed Operations facilities Using NASA's "GMSEC" Systems Architecture". *AIAA Space Ops 2008 Conference*, Heidelberg, Germany.