National Aeronautics and Space Administration

Smashing the Stovepipe

Leveraging the GMSEC Open Architecture and Advanced IT Automation to Rapidly Prototype, Develop and Deploy Next-Generation Multi-Mission Ground Systems

Presenting:

Paul Swenson

ASRC Federal Space & Defense

paul.swenson@nasa.gov

NASA Goddard Space Flight Center

Software Engineering Division

GSAW 2017

Session 11E: Adopting Agile Ground Software Development

Wednesday, March 15th, 2017

Los Angeles, California

Introduction

- Satellite/Payload Ground Systems
  - Typically highly-customized to a specific mission's use cases
  - Utilize hundreds (or thousands!) of specialized point-to-point interfaces for data flows / file transfers
- Documentation and tracking of these complex interfaces requires extensive time to develop and extremely high staffing costs

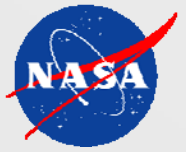

Photo © Warren Mars

- Implementation and testing of these interfaces are even more cost-prohibitive, and documentation often lags behind implementation resulting in inconsistencies down the road

ITSec, IA and Operational Security (OPSEC)

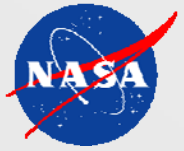
- With expanding threat vectors, IT Security, Information Assurance and Operational Security have become key Ground System architecture drivers
- New Federal security-related directives are generated on a daily basis, imposing new requirements on current / existing ground systems
  - These mandated activities and data calls typically carry little or no additional funding for implementation
- As a result, Ground System Sustaining Engineering groups and Information Technology staff continually struggle to keep up with the rolling tide of security

Multi-Mission Resource Sharing



- Advancing security concerns and shrinking budgets are pushing these large stove-piped ground systems to begin sharing resources
  - I.e. Operational / SysAdmin staff, IT security baselines, architecture decisions or even networks / hosting infrastructure
- Refactoring these existing ground systems into multi-mission assets proves extremely challenging due to what is typically very tight coupling between legacy components
- As a result, many “Multi-Mission” ops. environments end up simply sharing compute resources and networks due to the difficulty of refactoring into true multi-mission systems

Multi-Mission Resource Sharing (2)



- In many cases, Ground System baseline documentation was generated to the original “as-built” system
- Ground Systems continue to evolve post-launch
  - Changes not always captured in documentation
- When refreshing GS hardware, there is typically a complex “reverse-engineering” effort to derive the current state of software configurations and data flows so that existing capabilities can be fully re-implemented on the updated system
- CCB-tracked updates such as OS and software patches also muddy the waters, since these changes can alter the installation process and require additional steps to be taken

GMSEC Open Architecture

The Goddard Mission Services Evolution Center (GMSEC) project has worked to develop an open architecture for Ground System messaging

- Under development since 2001, utilized operationally since 2005
- Facilitates interoperability across GS Components via standard messaging protocol and industry-standard middleware options
- Promotes High (Functional) Cohesion across components
  - GMSEC-based components contribute to a specific well-defined task (i.e. TT&C, Alerting, Event-driven, Time-based or Product/File-based automation, Mission Planning, Situational Awareness / Visualization)
  - This allows for swapping of GS components or message routing middleware without requiring extensive interface-related NRE or any changes to existing or new components
  - For aging legacy ground systems, being able to pull out a no-longer-supported component and replace it cleanly with a modern equivalent is essential, and the GMSEC approach greatly simplifies this!

GMSEC Open Architecture (2)

- Provides for Loose Coupling between components
  - GMSEC messages provide a standards-based messaging approach that co-ordinates how components implement cross-system event logging, telemetry, command, control directive and status messaging
  - Greatly reduces the need for custom-built GS interfaces
  - Messaging specification is extremely extensible and is maintained at several different layers allowing for maximum flexibility
- Many benefits realized by these simplified interfaces:
  - Reduces complexity of documentation and architecture
  - Eases testing costs due to simplified interfaces
  - Helps facilitates the introduction of secure federated enterprise environments by providing a common message-based data fabric that can be extended across mission boundaries and Ground System components
  - Provides enhanced situational awareness capabilities

GMSEC API

- GMSEC API provides the interface that all components utilizes to publish / subscribe to messages on the bus
- Completely abstracts communications with the middleware
- Able to take advantage of native middleware security features such as transport-level encryption, source/destination-based controls
- GMSEC API available as open-source on SourceForge
- CompatC2 “Secure” API (available for Government use) layers on additional message-level security features such as payload encryption, message signing, message authentication and non-repudiation
- Active collaboration between NASA and other government space organizations

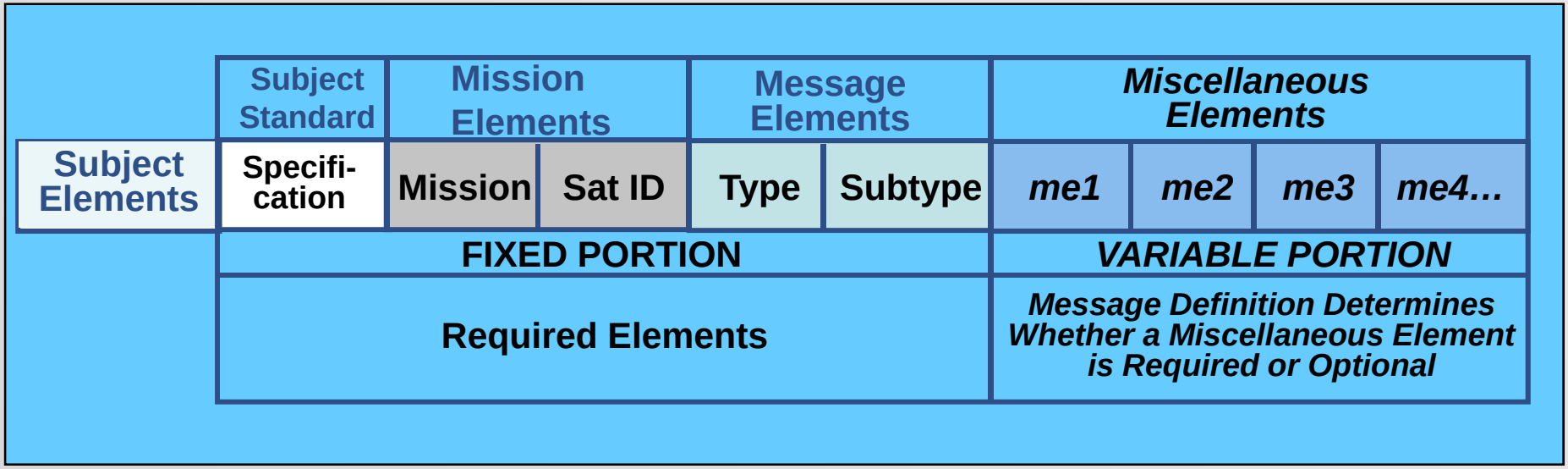
GMSEC Message Specification Governance

- GMSEC employs a 3-layer governance model for messages:
  - Local Level
    - Missions can develop their own usage document
    - Local naming conventions
    - Values for header fields
    - Selection of messages that will be used
  - CompatC2 Level (or other organization, NOAA, etc.)
    - Addendum generated for items of value across the DoD (Aerospace Corp. / Chantilly)
      - Satellite Naming Conventions
      - DoD-specific navigation message
      - General guidelines
  - GMSEC Level
    - NASA maintains the primary message specification volume
    - Available to interested groups upon request
- Good recent examples have demonstrated that this process works well.

GMSEC-Level Message Specification

- Messages at the Local Level or CompatC2 level are routinely reviewed for general applicability and if more universally useful, promoted to the next level
- The NASA GMSEC Message Specification provides a common set of message types for typical Ground System communications needs:
  - Event Messages
  - Component Directives
  - Framed/Packetized telemetry
  - Framed/packetized commands
  - Decommutated mnemonic messages
  - File/Product arrival messages
  - Etc.
- Each class of messages is published to a unique “message subject” that identifies the type of message and some key parameters describing the data contained therein
- This is also the mechanism that components can select which data to subscribe

GMSEC Message Subjects



Telemetry Message Subject Example:

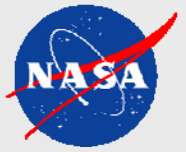
GMSEC. EOS .TERRA . MSG.TLM *.TAC.RT.CCSDSFRAME.2.1*

Fixed, required portion

Message dependent, variable portion

(Body of the message follows the above header)

GMSEC API 4



- Recent release of GMSEC API 4 provides a brand new re-designed interface to GMSEC
  - Leverages modern innovations in programming theory
  - Adopt best practices for modern object-oriented APIs
  - Streamlines integration of new components onto the bus and reduces coding errors
- GMSEC team has been working closely with mission and industry to adopt cutting-edge best practices into their development lab and engineer the building blocks for ground systems of the future
- Adapting an existing to GMSEC requires building a small adapter glueware for each message type
  - This typically can be done in a day or two even for a large and complex piece of software
  - The adapter code needs only to translate between the component's native interfaces and the GMSEC messaging protocol

Innovations in System Deployment

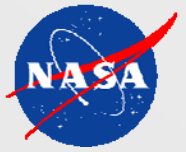
Recent missions and GMSEC initiatives have utilized Advanced IT automation tools to speed system implementation and deployment of new systems

- Rapid System Deployment Tools such as:
  Cobbler / Vagrant / Vmware Templates / Microsoft Deployment Toolkit (MDT)
  - Installs a customized OS image within minutes in a fully-automated fashion
  - Can utilize different templates / build scripts to customize images based on requirements
- Software CM / System Baseline tools such as:
  Puppet / Ansible / Microsoft Group Policy
  - Automatically deploy custom software / baseline overlay on base system (CIS Benchmark, USGCB, DoD STIGs)
  - Install application software (GMSEC API, GMSEC components, Ground System software / TT&C, Mission Planning, etc.)
  - Install supporting tools / utilities (Matlab, Perl/Tk, other supporting modules)
  - Continuously Enforce baselines and configuration in support of Continuous Monitoring and DHS Continuous Diagnostics and Mitigation (CDM) initiatives

System Deployment Process Improvement

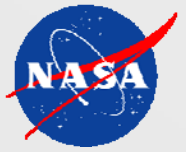
- System baselines and software installs are implemented as Puppet / Ansible code that is continually executed
  - Alternative would be paper set-up procedures which are hard-to-maintain and slow to utilize
- Configuration Managed code becomes your system baseline documentation
  - Ensures continual compliance with security policies which are audited and enforced on an hourly basis
- New system deployments take minutes instead of days
- Entire Ground System deployments can be scripted, iteratively tested, and most importantly re-produced as necessary

Change Management

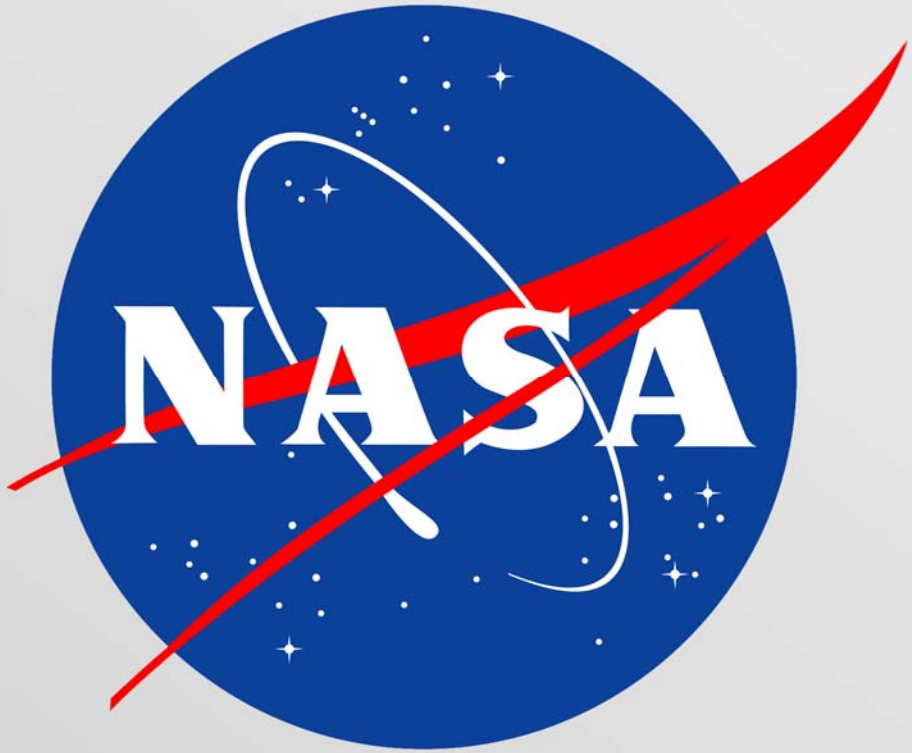


- Changes / updates to the ground system are also implemented via code / Group Policy
- This guarantees any newly-build systems will inherit the current up-to-date baseline
- Provides a boon to IT security thanks to the approach satisfying Continuous Monitoring directives with little or no extra effort
- Future multi-mission ground system components will install themselves from CM, configure themselves, install security baselines and be ready to operate within minutes
- This continuous integration approach to enterprise-level architecture and configuration management mirrors many of the approaches and techniques used in Agile development, and allows for a much more rapid, reproducible workflow for GS development and engineering

Conclusion



- Utilizing continuous integration / rapid system deployment technologies in conjunction with an open architecture messaging approach allows System Engineers and Architects to worry less about the low-level details of interfaces between components and configuration of systems
- GMSEC messaging is inherently designed to support multi-mission requirements, and allows components to aggregate data across multiple homogeneous or heterogeneous satellites or payloads
  - The highly-successful Goddard Science and Planetary Operations Control Center (SPOCC) utilizes GMSEC as the hub for it's automation and situational awareness capability
- Shifts focus towards getting GS to a final configuration-managed baseline, as well as multi-mission / big-picture capabilities that help increase situational awareness, promote cross-mission sharing and establish enhanced fleet management capabilities across all levels of the enterprise.


NASA