Integration and Test Approaches for Modern Ground Software Systems Workshop: What Works and What Doesn’t

Robert B. Crombie
Systems Integration and Test Office
Systems Engineering Division
Jason McKenney
Software Systems & Acquisition Department
Engineering and Technology Group

GSAW February 28, 2022: 9am-11:30amPT

Introductions

Participants' names, organizations, and recent ground I&T projects (please put in chat)

Outline and Timeline (next chart)

9:00-9:15am PT

Outline

Integration and Test (I&T) Approaches for Modern Ground Software Systems

- This workshop will allow participants to share their practical experiences, lessons learned, and best practices for the integration and test of software developed with modern practices
  - *We plan to capture bullets from the discussion on charts to recall the discussion*
- Timeline: (no scheduled breaks)
  - *Discussion by topic areas:*
    - 45 minutes on I&T at the **developer level**
      - *Where the software branches need to be merged into a software component and tested automatically,*
    - 30 minutes on I&T at the **system level**
      - *Where many components of the integrated ground system need to be demonstrated together,*
    - 45 min on I&T at the **system of systems level**
      - *Where the ground system needs to interoperate with other systems, such as the space system or a data processing system.*
- Discussion expectations:
  - *Topics will be introduced within each area/level above*
  - *If you have encountered the method/situation, please speak up and tell us any I&T lessons you learned, and best practices you have implemented*

Follow-up sessions may also be arranged if participants would like to explore particular topics in more depth

Modern Software I&T Topics Survey

What's in scope today

- Modern SW development practices: Have programs shifted to these?
  - *Agile software development*
  - *Continuous Integration/Continuous Delivery*
  - *DevOps and DevSecOps*
  - *Artificial Intelligence and Machine Learning*
- Modern Architectures and Deployment Techniques: Have programs shifted to these?
  - *Service Oriented*
  - *Microservices and API Gateways and Service Mesh*
  - *Data-Centric, Data As A Service*
  - *Event-Driven*
  - *Virtual Machines*
  - *Containers*
  - *Serverless*

I&T Experiences: Developer Level

The Developer Level is where the software branches are merged into a software component and tested automatically

9:15-10:00am PT

Challenging Areas We’ve Heard About – Developer Level

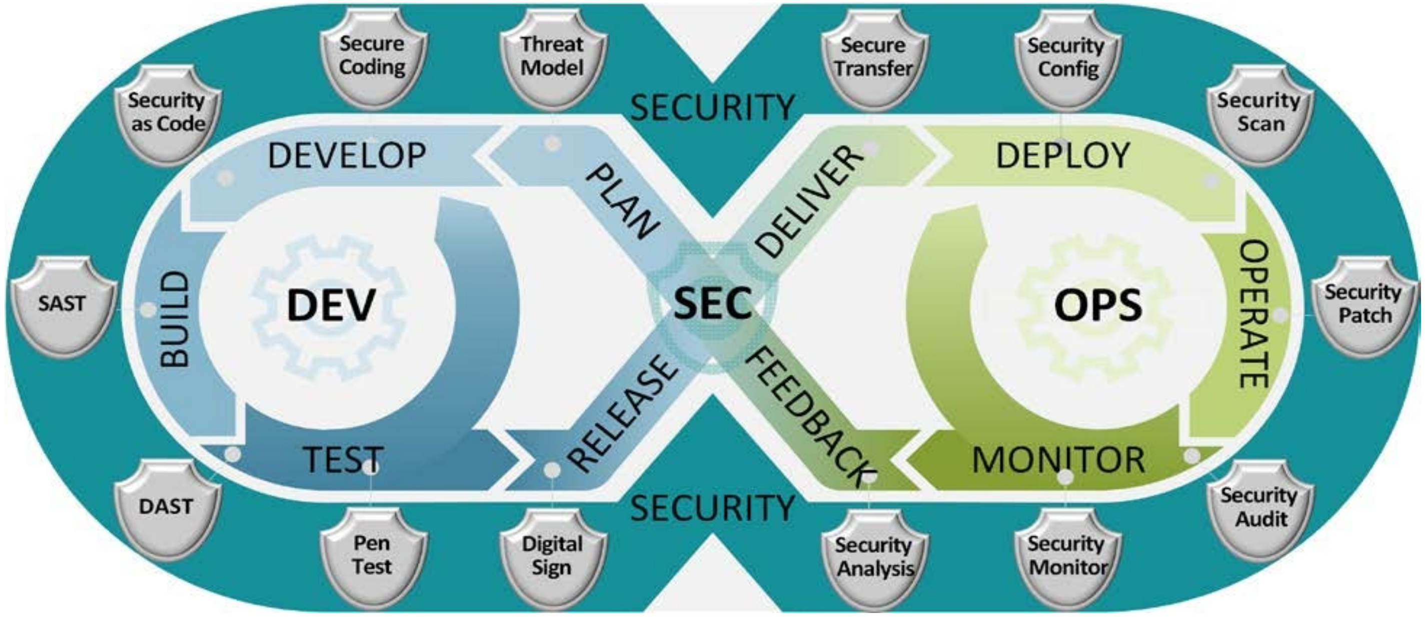
Comparing Traditional and Agile I&T Approaches

- Traditional vs Agile SW Development
  - *Which I&T Approaches are more effective?*
    - Behavior Driven Development (BDD)
    - Acceptance Test Driven Development (ATDD)
    - Test Driven Development (TDD)
  - *Unstructured with minimal planning?*
  - *Fixing errors mid-stream?*
  - *Reduced documentation?*
  - *How can MBSE help?*

Challenging Areas We've Heard About – Developer Level

Incorporating DevSecOps

- The DoD DevSecOps Lifecycle
  - How have you used it to improve Integration & Test? Best practices? Challenges?
  - How can it be used to support Scrum or other Agile development approaches?



DoD Enterprise DevSecOps Reference Design, 2019

Challenging Areas We've Heard About – Developer Level

Software Build and Deploy Process

- Are more programs shifting to microservices? What have been the results?
- What are some challenges with incorporating test automation into a continual software build process?
  - How often are builds done?
  - How are test cases grouped or managed?
  - What types of testing provide best results during the build/merge/deploy process?
    - Unit tests, static code analysis, test coverage analysis, automated acceptance tests, regression suites
- Branching and merging strategies for development teams?
  - Feature Branching, Release Branching, Trunk-based
  - How to handle patches or hot-fixes?
- User/operator involvement? At what stage?

Challenging Areas We’ve Heard About – Developer Level

Which Tests to Automate

- What Testing Should be Automated? Which have succeeded, which were challenging?
- Can automated tests cover all the important functionality without becoming too expensive?

- Have you encountered these Challenges for Automated Testing? How have you dealt with them?
  - *Test Traceability*
  - *Building a test automation framework*
  - *Keeping test runs light*
  - *Analyzing results*

Other Challenging Areas – Developer Level

Audience input

I&T Experiences: System Level

The System Level is where many components of the same ground system need to be demonstrated together.

10:00-10:30am PT

Challenging Areas We've Heard About – System Level

System integration of developed components

- Integration Challenges
  - *How can organizations address the rapid pace and increased integration and test frequency when CI/CD methods are used?*
  - *Have organizations found more frequent I&T of smaller changes to be cost effective?*
  - *Lessons Learned from service-based and API I&T between components?*

Challenging Areas We've Heard About – System Level

- Regression Testing challenges at the System level?
- Utilize canary testing or chaos to evaluate the software design?
- What sorts of tools or workflow processes work the best to support CM?
- How much User/operator involvement?
- What kind of Non-Functional Testing do you do at the System level?
- Continuously monitor performance & quality (in production)?

Other Challenging Areas – System Level

Audience input

I&T Experiences: system of systems

The system of systems level, where the ground system needs to interoperate with other systems, such as the space system or a data processing system.

10:30-11:15am PT

Challenging Areas We've Heard About – System to System Level

I&T with spacecraft or other ground systems, for example

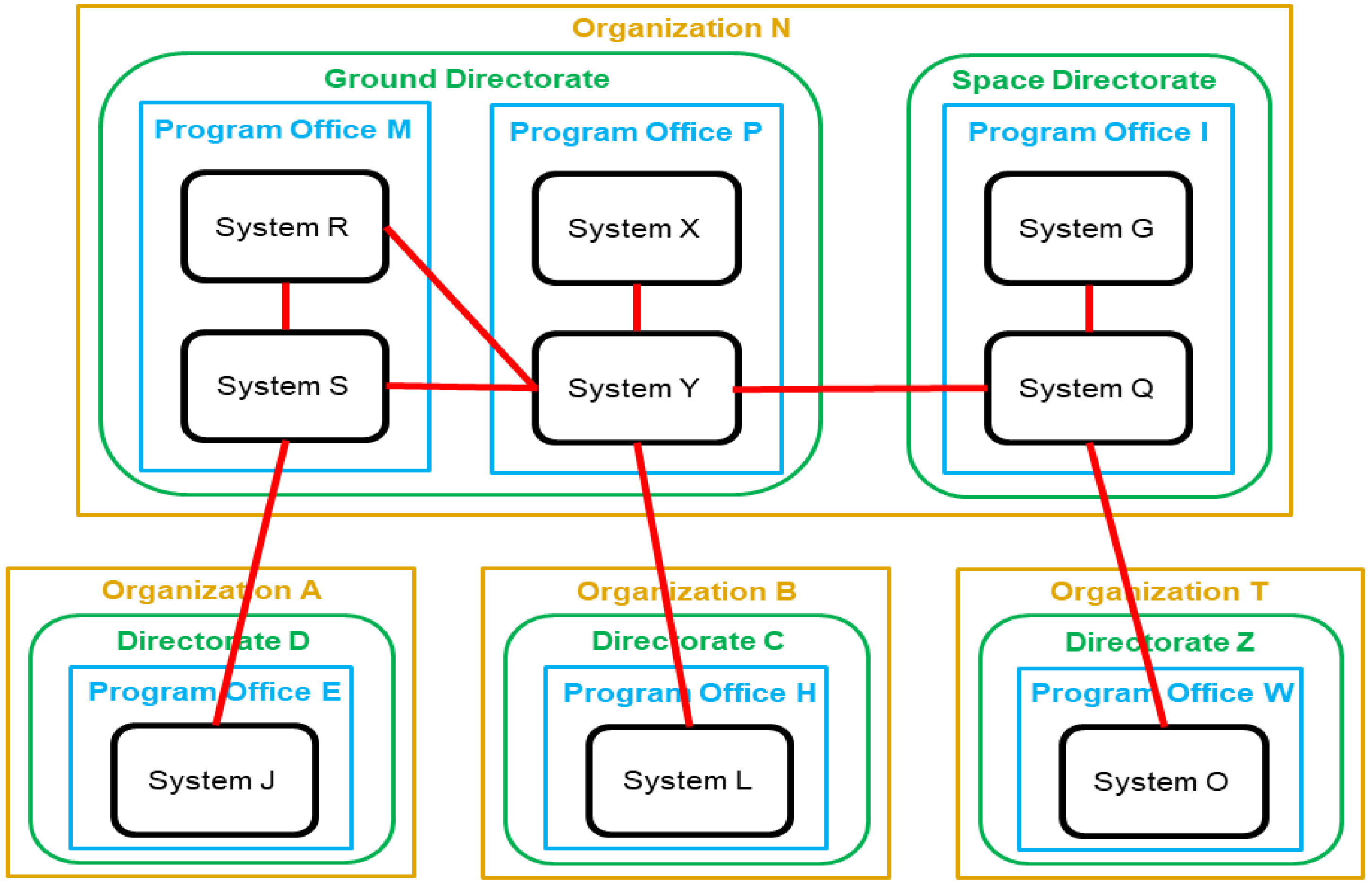
- Topics (subsequent charts) for discussion:
  - *Cross-Organizational Planning*
  - *Cloud to Cloud vs. Cloud to On-Premises*
  - *Other*

Cross-Organizational I&T Planning (1/2)

What we mean

- Organization to Organization
- Directorate to Directorate, same org
- Program Office to Program Office (PO)
- System to System, same PO

The most complex cross-organization test situation we've encountered:



Test objectives emphasize interfaces and interactions among systems from <u>different developers</u>

Cross-Organizational I&T Planning Challenges (2/2)

What is your experience?

- Roles & Responsibilities (who does what?) government vs prime vs integrating contractor?
  - *Who is your Interface design lead, integration lead, system-system test lead? Did it work well?*
- Scheduling
  - *Planning: how do you do rigorous planning while still **shifting left** for early testing to integrate faster?*
    - When do you start planning? How do you take advantage of modern SW Development methods?
  - ***How to set a test schedule** respecting agile development schedules of each system?*
  - *Common timeboxes to align system deployments? Incremental tests?*
- Test Resources
  - *Enterprise test environment or individual home environments for S-S tests?*
  - *Mod & Sim of other systems for virtual SoS?*
- Test data management for each mission thread
  - *Who develops it to be time-synched with the rest of the test environments?*
- Do (lower-level) system-level tests reduce (higher-level) system to system tests?
- User/operator involvement
  - *Users on console?*
  - *User Experience changes noticed due to modern methods?*

<u>Early</u> ***incremental testing of multiple systems together***

Who has done this during development? How did it work?

- Preplanning of increment content and schedule:
  - *How do you know what interfaces are ready for early testing between systems?*
- Establishing a viable test environment
  - *How do you connect the two systems for early interface and functional testing?*
    - Pull the other system’s latest software release into your test environment?
    - Establish a common (e.g., government) test environment?
- Test selected API interfaces (and not others) and system interactions
  - *Which threads do you selectively test?*
  - *Generation and use of customized test data*

Connections: What benefits and challenges have you experienced?

System B:	System A: On-Premises Infrastructure	System A: Homogenous (Single-Cloud)	System A: Heterogenous (Multi-Cloud)
On-Premises Infrastructure	Benefits: Challenges:	Benefits: Challenges:	Benefits: Challenges:
Homogenous (Single-Cloud) Infrastructure		Benefits: Challenges:	Benefits: Challenges:
Heterogenous (Multi-Cloud) Infrastructure			Benefits: Challenges:

Other Challenges of System-to-System Integration and Test

- **Cybersecurity** aspects: are all systems approved to operate (ATO)?
  - *Have you achieved continuous ATO? What were the challenges?*
- **Technology Stack** maintenance and synchronization (OSS, Libraries, services, APIs): Lessons learned?
- Use of **simulators/emulators** and keeping them current and validated: Lessons learned?
- **Interface** test challenges
  - *Message based (APIs) tests vs Mission/function based (threads) tests: Which work better?*
  - *API Gateways between systems? Experiences?*
- **First of a kind** (satellite system)-to-(ground system) testing:
  - *Has modern software development reduced the risk to initial on-orbit operations?*
  - *Can you reduce the testing needed for follow-on satellites?*
- Do your SoS tests verify or validate **non-functional and performance** requirements?

Other Challenging Areas – System to System Level

Audience input

Conclusions

Integration and Test Approaches for Modern Ground Software Systems:
What Works and What Doesn’t

11:15-11:30am PT

Overall Conclusions

Integration and Test Approaches for Modern Ground Software Systems: What Works and What Doesn't

- Developer Level
- System Level
- System to System Level
- Next steps

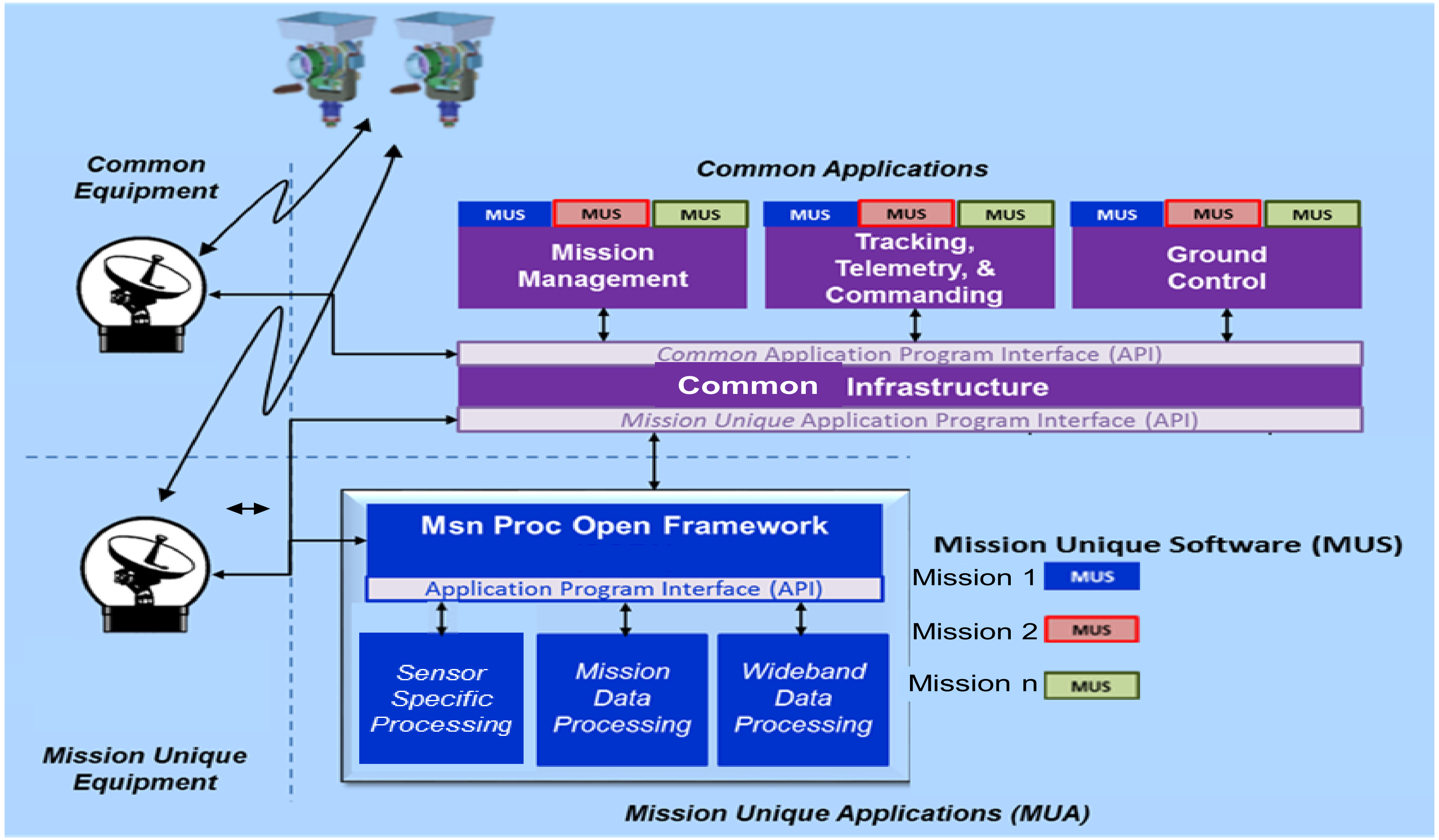
Backup

Testing Cloud to Cloud, or Cloud to On-Premises

Definitions

Infrastructure Model	Description
On-Premises (i.e., Private Cloud)	System deployed within on-premises **datacenter** or hardware (e.g., AUE). This has been the traditional approach to deploying systems.
Homogenous Cloud / Single-Cloud	System deployed within a **single** cloud infrastructure (e.g., AWS)
Heterogenous / Multi-Cloud	System deployed across **different clouds** (e.g., some components may be deployed on Azure and others on AWS.
Hybrid	System deployed using a combination of **on-premises** and one or more **cloud**(s) (e.g., some components are deployed on premises while others are deployed in one or more clouds such as Azure, AWS).

Generic Ground Segment Architecture



Common Equipment
Common Applications
MUS
MUS
MUS
Mission Management
Tracking, Telemetry, & Commanding
Ground Control
Common Application Program Interface (API)
Common Infrastructure
Mission Unique Application Program Interface (API)
Msn Proc Open Framework
Application Program Interface (API)
Sensor Specific Processing
Mission Data Processing
Wideband Data Processing
Mission Unique Software (MUS)
Mission 1 MUS
Mission 2 MUS
Mission n MUS
Mission Unique Equipment
Mission Unique Applications (MUA)

Modern Software Development Techniques/Architectures/Environments

What we mean by "modern"

- Modern **development techniques** may include:
  - *Agile software development*
  - *Model-Based Systems/Software Engineering (MBSE)*
  - *Continuous Integration/Continuous Delivery (CI/CD)*
  - *DevOps, DevSecOps*
  - *Test Driven Development (TDD)*
  - *Artificial Intelligence/Machine Learning (AI/ML)*
- Modern software system **architectures** may include:
  - *Service Oriented Architectures (SOA)*
  - *Microservices, Service Mesh*
  - *Event Driven Architecture*
  - *Data-Centric Architecture, Data Lakes*
- Modern development, test, and deployment **environments** may include:
  - *Cloud Services (IaaS, PaaS, SaaS, serverless) vs On-Premises (private cloud)*
  - *Multi-Cloud, Hybrid cloud*
  - *Infrastructure as Code (IaC)*
  - *Containers*

If you have integrated and/or tested software using any of the above approaches, please share your experiences