Ground System Agile Software Development and Mission Assurance

Jermaine Brinson
Mission Assurance Strategy Office

January 2026

Approved for public release. OTR 2026-00254.

Agile Ground System Development

- Agile Ground Development is marketed as the “holy grail” of ground system engineering
- But how is it going in real life?
- What follows are some takeaways from recent discussions on agile ground projects

Ground System Agile Development Major Themes

- Boots on the ground
  - Mission assurance requires more hands-on direct engagement for agile - no big “sell-off” event
  - User involvement is critical
- Ground and space go hand in hand
  - For ground to be agile, space must be agile
  - Space systems are rarely modular, flexible, developed in sprints, etc.
  - Must improve edge deployment
  - Hardware and software schedule dependencies make operator involvement and testing difficult
- Contracting pathways and acquisition processes are not set up for agile
  - System behaviors vs. requirements
  - Who owns the roadmap?
  - To MBSE or not to MBSE?
  - Milestones need more granularity and well-defined expectations (not just IOC or FOC)

Ground System Agile Development and MA Major Themes

- Schedule and resource estimation and management
  - Can’t do this for waterfall, let alone for agile
  - What tools exist for functional point estimation?
- Critical “ilities”
  - Scalability, flexibility, modularity, etc.
  - How do we measure and assess non-functional requirements (like flexibility, modularity, etc.)?
- Testing, testing
  - Never enough test assets (e.g., software simulators for the space vehicle)
  - Consider greater use of on-orbit test and checkout assets and operations team members
  - Need better automated regression testing
  - Developmental and operational testing must be pulled to the left
  - Mission assurance requirements testing and sell-of should be part of the “definition of done”

Ground System Agile Development and MA Major Themes

- The metric system
  - Metrics are hard to define, subject to fluctuation and misinterpretation, and can drive bad behavior
  - Metrics are good at shining a light on the problem of the day, but lose attention once the problem is solved
  - Need a common-sense approach to measuring progress against metrics
- Continuous integration / continuous delivery pipelines and software factories
  - Several types of implementation: “homegrown,” commercial, government (Platform One)
  - All can enable rapid development, but all have pros and cons
  - What case studies exist?
- Cybersecurity and Authorities to Operate
  - Needs to be baked in from the beginning
  - But external approvals are complicated

Ground System Agile Development and MA Major Themes

- The humans (and processes and business cases) in the loop
  - Team capabilities must be high
  - Team must be familiar with Agile (or trained on it)
  - Incentivizing modularity is hard
  - Software-centric company vs. space-centric company: must balance technical capability with practical experience
  - Professional services: use the help they give you, but recognize they can’t solve everything
  - Don’t depend on a single vendor

MA Framework

Key Tenets of an Agile Ground SW Development MA Framework

- Establish a Mission Assurance-Driven Agile Culture
- Iterative Development with Incremental Assurance Gates
- Incorporate DevSecOps Practices
- Risk-Based Testing and Validation
- Transparent Metrics and Adaptive Governance
- Pilot and Scale Gradually

Establish a Mission Assurance-Driven Agile Culture

- **Integrated Cross-Functional Teams:**
  Form teams that include not only developers and testers but also mission assurance experts, security engineers, and compliance professionals. Their continuous involvement ensures that risk and safety requirements are built into every sprint.

- **Agile Roles with Assurance Responsibilities:**
  Define roles such as a “Mission Assurance Champion” or “Agile Quality Advocate” within teams. These roles are responsible for monitoring that every user story includes clear acceptance criteria related to mission assurance—from performance and reliability to security and compliance.

- **Mission Assurance Backlog Items:**
  Incorporate risk identification, mitigation tasks, and quality criteria directly into the backlog. Each story should have defined “Mission Assurance Acceptance Criteria” that are reviewed and refined in backlog grooming sessions.

Iterative Development with Incremental Assurance Gates

- **Definition of Done Expansion:**
  Expand the definition of done to include mission assurance checkpoints such as:
  - *Automated security scans*
  - *Performance and stress tests*
  - *Compliance verification*
  - *Risk assessment updates*
    This ensures that each sprint deliverable meets both functional and critical assurance standards before moving forward.

- **Incremental Assurance Gates**

- **Continuous Integration and Continuous Testing (CI/CT):**
  Adopt a robust CI/CT pipeline where code changes automatically trigger unit, integration, and system-level tests designed to validate both **functionality** and **mission assurance** requirements (including security, reliability, and performance).

Incorporate DevSecOps Practices

- **Automated Security and Compliance Testing:**
  Include automated scanning tools (e.g., static code analysis, dynamic vulnerability testing) as part of the CI pipeline. This integration makes security an automated, non-negotiable part of the development process.

- **Drift Detection & Configuration Management:**
  Use automated configuration and infrastructure-as-code tools to ensure that deployment environments are secure and consistent. This minimizes risks linked to environment drift and unplanned configuration changes.

- **Continuous Monitoring and Feedback:**
  Implement real-time monitoring tools for deployed systems to capture operational data (performance, incident logs, security events). Feedback loops must be integrated into agile reviews to adjust priorities or defenses as needed.

Risk-Based Testing and Validation

- **Risk-Tiered Test Strategy:**
  Prioritize test cases and validation efforts based on the criticality and risk impact of different software components. High-risk features undergo additional layers of testing (e.g., stress tests, fault injection, and scenario-based testing).

- **Simulation & Modeling:**
  Use simulation environments that replicate mission-critical conditions. Continuous, iterative testing in these simulated conditions allows teams to build confidence in the system under scenarios that mirror real-world operational stress.

- **Periodic Independent Reviews:**
  Schedule regular independent assurance audits—either internal or external—to evaluate adherence to mission assurance objectives. These reviews act as an additional safeguard, lending an unbiased perspective on risk posture.

Transparent Metrics and Adaptive Governance

- **Agile Metrics Coupled with Assurance Indicators:**
  Monitor traditional agile metrics (velocity, sprint burndown) alongside dedicated mission assurance indicators. These might include defect escape rates related to security vulnerabilities, regression test pass rates, or time-to-resolve identified risks.

- **Adaptive Governance Framework:**
  Develop an adaptive governance model that permits flexibility when emerging risks (or opportunities to enhance assurance) are detected. Regular program-level retrospectives can help adjust processes, update risk profiles, and refine the DoD's mission assurance strategy.

- **Incorporate Lessons Learned:**
  Use every sprint retrospective to capture insights related to assurance issues. Incorporate these lessons into planned improvements, ensuring that the framework evolves based on real-world feedback.

Pilot and Scale Gradually

- **Start with Low-Risk Pilots:**
  Implement the framework on a smaller, low-risk project to validate processes, tools, and cross-functional collaboration. Use these pilots to refine assurance criteria and agile ceremonies before scaling to larger, mission-critical systems.

- **Iterative Scaling and Roadmapping:**
  As confidence grows, incrementally expand the framework to cover more components and larger teams. Maintain a clear roadmap that aligns agile iterations with long-term mission assurance objectives.

Summary/Key Take Aways

- **An agile mission assurance framework for ground system software development should be designed to meet the rigorous demands of operational reliability, performance, security, and compliance while leveraging the speed and flexibility of agile and DevSecOps methodologies.**

- **By embedding mission assurance into every phase of the agile lifecycle, organizations can deliver high-quality, mission-ready ground systems that adapt responsively to emerging challenges and requirements.**

“50 (ish) Questions to Ask”

Questions to Ask - 1. Contracting Stage

- 1.1. Do contract language and performance metrics explicitly require adherence to Agile processes, and do they allow for iterative, cloud-based development approaches?

- 1.7. How are risks identified, allocated, and managed within the contract, and what provisions exist to encourage innovation and experimentation throughout the project?

Questions to Ask - 2. Planning and Requirements Stage

- 2.4. How is traceability established between requirements, user stories, and “definition of done” acceptance criteria? Do we have a clear, flexible product roadmap with prioritized features, and how are decisions around these priorities made?

- 2.6. How will mission assurance requirements be incorporated into the Agile cycle and sprint planning?

Questions to Ask - 3. Design Stage

- 3.1. Have interfaces between non-agile legacy systems been clearly identified with dependencies documented, including those that might constrain cloud integration and commercial system interoperability?

- 3.8. How does the design incorporate risk mitigation strategies for technical debt, plans for “on/off-ramps?”

Questions to Ask - 4. Development and Iteration Stage

- 4.1. What mechanisms are in place to validate that each iteration's requirements remain aligned with program priorities and risk thresholds?

- 4.3. How is compliance with mission assurance and cybersecurity requirements enforced during each agile development cycle?

Questions to Ask - 5. Verification & Validation (V&V) Stage

- 5.1. Is there a comprehensive V&V plan that details how requirements, user stories, and acceptance criteria will be integrated, verified, and validated within every sprint?
- 5.3. How are automated test suites configured to continuously verify both the functional performance and mission assurance criteria (e.g., security, reliability, usability) during agile iterations?

Questions to Ask - 6. Deployment & Sustainment Stage

- 6.1. What is the rollback or remediation plan if an agile increment fails to meet requirements or mission assurance criteria post-deployment?

- 6.2. How are continuous monitoring tools and dashboards employed to track operational performance, security posture, and risk indicators after each deployment?

Questions to Ask - 7. Process Improvement & Feedback Stage

- 7.1. How are lessons learned from each sprint and integration cycle systematically captured and translated into process enhancements?

- 7.2. What formal feedback loops exist between operations, development, and mission assurance teams to refine performance metrics, risk tolerances, and interface management approaches over time?

Additional Considerations

- Ensure that agile practices are tailored to both the unique challenges posed by government requirements and the rapid innovation seen in commercial approaches (e.g., “fail fast, learn fast”).

- Continue to refine contract, planning, and execution strategies as cloud migration becomes more prevalent, ensuring that policies facilitate seamless transitions between legacy approaches and new agile methodologies.

- Embed security and configuration management early in the design and development cycle to mitigate emergent risks associated with broader technology adoption and cloud-based operations.