The Lean Agile Architecture

Ms. Lisa Ann Henke
Chief Architect Global Support Programs

Mr. Matthew Reider, SAFe SPC
DevSecOps Engineer
Introduction

- Focus on WHY
- Words of Betty Sapp, Lower the ratio between what she calls the "doers" and the "reviewers"
- Role of Architecture and Systems Engineering in an agile construct (not 25 DoDAF artifacts for delivery and review)
- Acknowledgement that not all projects and programs require the same level of rigor. Important to evaluate what level of rigor is necessary
Focus on Why

- Maximize the amount of work not done
- System Engineering/Architecture artifacts and documentation should have a known audience and purpose for the communication
  - Ask:
    - Who is it for?
    - What is its purpose?
    - When is it needed?
    - Why am I generating it?
    - How can I automate it?
- Balance top down systems engineering with emergent design through the use of the agile cadence

"We have some of that same urgency in terms of where adversaries are going both in their own space-based capabilities and in their counter-space," she explained. "But we don't quite have that same level of risk tolerance or the processes that support a lot of going fast."

She cited the need to lower the ratio between what she calls the "doers" and the "reviewers," the ballooning set of overseers in the Pentagon and on Capitol Hill that have slowed the acquisition process.

"How this all comes out, I don't know," she said of the reform push. "To just reorganize the doers, I think, we will have fallen short. A lot of what's in the way of the doers is the processes above them."

~ Politico Article 'We are getting left behind'
The nation's spy satellite chief calls for empowering the 'doers' by reducing the 'reviewers.'
By BRYAN BENDER 08/17/2018 06:49 AM EDT
Lean Agile Architecture

- Lean Six Sigma and Agile development overlap
  - Both focus on removing waste and eliminating/reducing defects
  - SAFe, LeSS, and DAD have "Systems Engineers/Architects," "Architecture and Design," and "Architecture Owners" described but how and what they do is left somewhat undefined.
    - The Scaled Agile Framework for the enterprise (SAFe),
    - Large Scale Scrum (LeSS)
    - Disciplined Agile Delivery (DAD) SSAFE

Major Agile methodologies recognize the role of Architecture and SE
Agile with Lean Systems Engineering

- Scaled Agile Framework for the Enterprise
  - "The Systems Engineering and Architect are responsible for defining and communicating a shared technical and architectural vision."
    - Evolves over time while supporting needs of current users
  - Avoids overhead and delays associated with phase-gate and BUFD- methods
    - Ensures the ‘system always runs’
    - Balances emergent design and intentionality
    - Takes a systems view across the full value stream
Agile with Lean Systems Engineering

- **Large Scale Scrum (LeSS)**
  - "The only software documentation that actually seems to satisfy the criteria of an engineering design is the source code" (Jack Reeves, "What is Software Design")
  - First observation –The sum of all the source code is the true design blueprint or software architecture.
  - Second observation –The real software architecture evolves (better or worse) every day of the product, as people do programming.
  - Third observation –The real living architecture needs to be grown every day through acts of programming by master programmers.
  - Fourth observation –A software architect who is not in touch with the evolving source code of the product is out of touch with reality ("Powerpoint Engineer.")
  - Fifth observation –Every programmer is some kind of architect—whether wanted or not. Every act of programming is some kind of architectural act—good or bad, small or large, intended or not.
Agile with Lean Systems Engineering

- Disciplined Agile Delivery
  - The "Identify Architecture Strategy" process occurs early in program kickoff and then prove that architecture early in development
    - It enables effective evolutionary architecture.
    - We want to identify, and hopefully eliminate architectural key risks early.
    - Avoid technical debt.
    - Improved DevOps integration.
    - Enables us to answer key stakeholder questions.
    - Enhance initial scoping and planning efforts.
REALM (Rapid Enterprise Agile Lean Methodology)

● "Intentionality"
  ○ We incorporate a time-boxed up-front design session to establish an architectural vision delivered with the program vision
  ○ We maintain (preferably automated) lean architectural views of the system at the level necessary for the system in development

● "Emerging"
  ○ We incorporate an iterative design review at the beginning of each Program Increment or Sprint planning to discuss, decide, and update the architecture with the entire team, in a room full of whiteboards
  ○ We automate documentation from the codebase, we do not "powerpoint engineer" a set of documents that are given to the developers
Logical Backlog Decomposition – Concept to Deployment in Every Sprint & Release

Release Planning
Every three sprints, the RTE leads the release planning session defining Features and Release Goals.

Feature Backlog Refinement
Wednesday prior to Sprint Planning, Business Owners collect Feature priorities, give to PO.

IDR
Prior to Sprint Planning, SE/SA lead architectural discussions to support features and stories.

Story Backlog Refinement
Weekly, the PO/SM generate stories that support Feature completion.

Sprint Planning
Features, Sprint Backlog, Architecture brought to team as “ask.” RTE/SE and Team discuss, assign, and point all stories. When approved by team, sprint begins.

CCB
Proposed changes in architecture, software additions, and security relevant changes are brought to ISSE, reviewed and approved after planning.

Current Sprint

Demo Planning
Prior to end of Sprint, user stories marked as “demo-ready” are used to generate demo agenda. SM leads.

Review/Demonstration
All work completed by dev team shown by developers to other developers and customers.

Build/Merge Cycle
All work completed during sprint is merged into release trunk by one or more team members with regular mob merges.

Deployment
Completed Sprint code deployed to dev and stage environments at customer site by dev team, automated tests run.

Release Deployment
After successful test event, completed Release code deployed to production by dev team.

CI/CD-TDD
Code is automatically tested at every check-in.

Daily Scrum
Ticket status, progress, and “blockers” discussed daily. SM coordinates with SE team track and eliminate blockers.

Entire Team - Sprint

All Teams - Release Only

SM/PO and SME - Sprint

Cadence
4 week sprints, 12 week releases, Time boxed development
Agile and Lean Systems Engineering Methodology

Requirements
Align requirements at spin planning – ID reqs under test and full or partial sell off

Architecture
Iterative Design Review (IDR) held with Sprint Ceremonies, used as a readiness evaluation for features, & builds the Body of Evidence

Design
Reviewed at IDR
Intent is to maximize sell off
Updated during Backlog Grooming & Sprint Planning

Construction
Every Sprint is sacred.
Artifacts updated to document implementations.

Integration & Validation

Function Test
Automated testing of code

Integration Test
Automated testing provides integration and regression testing at every build

Acceptance Test
Execution aligned w/ Spin Demo
Witnessed testing to plan
Event analysis and reporting
(Delivery report 1 week past demo)

Unit Test
All code unit tested and deployed nightly
Code automatically built, deployed as containers on a PaaS

Unit Test
All code unit tested and deployed nightly
Code automatically built, deployed as containers on a PaaS
Architectural Vision (a.k.a Design Guardrails)

- Agile defines a "Program Vision" that discusses what the software will do for what users.
- REALM includes an "Architectural Vision," described by the Open Group Architecture Framework, (TOGAF) as "essentially the architect's "elevator pitch" - the key opportunity to sell the benefits of the proposed development to the decision-makers within the enterprise"
- Intentional, high level architectural goals that emerging architectural decisions will align to and be challenged against

"Ruthless Automation"
- Push button delivery
- Standard software tools to create, manage, and execute
- Scripted code integration,
- Unit and functional testing,
- Documentation updates,
- Environment & software deployments,
- Software configuration

"Platform Neutral"
- Use of Platform as a Service (PaaS) container management to alloy deployment of services to various environments without extensive rework
- Allows individual services to be automatically added, scaled, versions, or distributed on any infrastructure

Web Service Communications
- Standardized, stateless, web-based RESTful service APIs are available to internal or external consumers
- Open Geospatial Consortium (OGC) compatible services allow use of services by other systems

Reusable, Context-Bound Microservices
- Individual services built, added, versioned, or retired without extensive rework
- Configurable coordination of services into processing chains or applications
- Allows for individual services to be reused by multiple interfaces or external consumers
Emerging Architecture Methodology

**Step 1: Determine Intended Use of Architecture**
- Align requirements at sprint planning – ID reqs under test and full or partial sell-off

**Step 2: Determine Scope of Architecture**
- Reviewed at IDR
- Intent is to maximize sell-off
- Updated during Backlog Grooming & Sprint Planning

**Step 3: Determine Data Required to Support Architecture Development**
- Every Sprint is sacred. Artifacts updated to document implementations.

**Step 4: Collect, Organize, Correlate, and Store Architectural Data**
- Deployed and built, developers on a PaaS

**Step 5: Conduct Analyses in Support of Architecture Objectives**
- Automated testing provides integration and regression testing at every build

**Step 6: Document Results in Accordance with Decision-Maker Needs**
- Execution aligned with Spin Demo

**Requirements**
- Align requirements at sprint planning – ID reqs under test and full or partial sell-off

**Acceptance Test**
- Execution aligned with Spin Demo

**Integration Test**
- Automated testing of code

**Function Test**
- Automated testing provides integration and regression testing at every build

**Construction**
- Every Sprint is sacred. Artifacts updated to document implementations.
Ruthless Automation – Automate Documentation

Tools reflect the true state of the “as-is” development and not someone’s interpretation of development status.
Lessons Learned

• You have to know the WHY to communicate the WHY

• Acknowledgement that not all projects and programs require the same level of rigor. First important step is to evaluate what level of rigor is necessary

• Ruthlessly automate – make the tools work for you
Questions?

Ms. Lisa Ann Henke
Chief Architect Global Support Programs
lisa.henke@radiantsolutions.com
703-947-8313

Mr. Matthew Reider, SAFe SPC
DevSecOps Engineer
matthew.reider@radiantsolutions.com
703-947-8312