Smarter Acquisition with Agile Approaches

Session 11f

Supannika Mobasser and Jodene Sasine
The Aerospace Corporation

Approved for public release. OTR 2019-00286

Overview

- *Agile software and system development is no longer a new topic for the Government sector.*
- *Significant challenges to employing Agile methods as typically applied in the commercial software-intensive industry.*
- *An additional challenge is how to smartly apply Agile concepts, not only to the software system development, but to the whole ground system acquisition life-cycle.*
- *Discussion topics*
  - ***Smarter software factory and product delivery***
  - ***Smarter program oversight and incentive structure***
  - ***Smarter quality assurance, compliance, and accreditation***
  - ***Smarter practices and other domains***

- *Share your Agile adoption experiences and learn from others*
  - *Participants with all levels of Agile expertise are welcome.*

Introduce ourselves

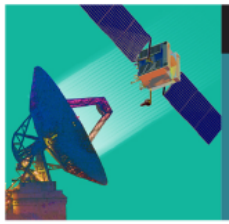
- What is your name?
- Where are you from?
- One good thing about your experiences in Agile adoption
- One pain point about your experiences in Agile adoption
- What's your expectation about this working group?

Schedule

Time	Presentation and Discussion
1:00 – 1:20pm	Session Overview
1:20 – 1:45 pm	Agile Working Group 2018 Outbrief Jodene Sasine, The Aerospace Corporation
1:45 – 2:10pm	Scaled Agile in a traditional fixed contract world: A case from Satellite Monitoring and Control Enrique Fraga Moreira, GMV Aerospace and Defence
2:10 – 2:35pm	Revisit on Agile Fit Check Supannika Mobasser, The Aerospace Corporation
2:35 – 3:00pm	Agile Anti-Patterns Supannika Mobasser, The Aerospace Corporation
3:00 – 3:30pm	Break
3:30 – 5:00pm	General discussion • ***Smarter software factory and product delivery*** • ***Smarter program oversight and incentive structure*** • ***Smarter quality assurance, compliance, and accreditation*** • ***Smarter practices and other domains***

Working Group Outbrief

Ground System Architectures Workshop



Session 11D

Achieving Resiliency with Agile Methods

Supannika Mobasser and Jodene Sasine,
The Aerospace Corporation

Approved for public release. OTR 2018-00491

GSAW 2018 Outbrief

Ground System Architectures Workshop

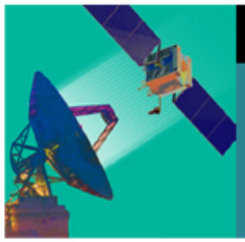
Session 11D

Participants

- Lauren Ballard, Nesbitt Discovery Academy
- Lyle Barner, JPL
- Emily Brison, Nesbitt Discovery Academy
- Doug Buettner, Aerospace
- Jay Bugenhagen, NASA
- Brook Cavell, Aerospace
- Roger Claypoole, Aerospace
- Eric Cohen, Lockheed Martin
- Enrique Fraga, GMV
- Judy Kelley, ASRC Federal
- Peggy Lou, Aerospace
- Paul Mallon, Aerospace
- Ugur Melihslizue, TAI
- Sue Mobasser, Aerospace
- Phuong Phan, Navy
- Jodene Sasine, Aerospace
- Jim Schier, NASA
- Scott Smith, SAIC
- Bruce Steiner, Aerospace
- Michael Thimblin, Aerospace
- Rolando Ventura, Harris
- Russ Wolfer, USG

GSAW 2018 Outbrief

Ground System Architectures Workshop



Session 11D

Schedule

Time	Presentation and Discussion
1:00 – 1:30pm	Session Overview
1:30 – 2:00pm	"Agile ground segment software development: cheaper, faster and better" Enrique Fraga Moreira, GMV Aerospace and Defence
2:00 – 2:30pm	"SCRUB for Peer Review of Static Code Analysis Results" Lyle Barner, Jet Propulsion Laboratory
2:30 – 3:00 pm	General discussion - I • ***Agile Battle Rhythm :*** *who, what, when, where, why, how many*
3:00 – 3:30pm	Break
3:30 – 5:00pm	General discussion – II • ***Agile Architecture:*** *build "-ilities" and resiliency in* • ***Agile Enterprise:*** *cultural and paradigm shift* • ***Agile Mission Assurance:*** *trust but real-time verify* • ***Agile Supporting Infrastructure:*** *required product and process resources*

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Battle Rhythm

- **Who?**
  - **Default:** Scrum Product Owner, Scrum Master, Developers and Testers
  - Team composition? Any special team, such as system engineering team, integration team, program management, customer liaison, Integrated Product (Process) Team (IPT)?
    - Embed Architect, SMEs, Requirements Engineer, SE into Agile team
    - Developers knowledgeable on cybersecurity or have a dedicated Cyber engineer
    - Release Train Engineer – manage multiple teams, release tempo
    - Human Factors Engineer – overall program, involve user communities
  - Who is your Product Owner?
    - Government – requires training/education; challenging due to frequent Govt rotation (consider if 2-3 month overlap is possible for cross training)
    - Contractor – ensure regular communication between Customer and Product Owners
    - Contractor Product Owner needs Govt counterpart to synchronize
  - Required certifications for Product Owner? Scrum Master?
    - Must be experienced

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Battle Rhythm

- **What?**
  - **Default:** Sprint Planning, Daily Stand-up, Sprint Demo, Sprint Retro, Story Grooming?
    - Scrum of scrums
    - Pre-release / Post-release (build / increment / iteration) Reviews
  - How to collaborate across teams?
    - Utilize Release Train Engineers
  - Any additional / tailored activities for the new roles?
    - Govt counterpart to Contractor Product Owner
    - Govt engineer/developer embedded/deployed into Contractor Agile team that Govt pays for

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Battle Rhythm

- **When?**
  - Sprint length? Release length? Number of Sprint per Release?
    - Sprint length: 2-4 weeks
    - Release length:
      - Greenfield: quarterly (minor); 6 months (major)
      - Enhancements: monthly
      - Stakeholders constrained: 9 months
    - Number of sprints per release: depends on release length
  - Any empty/buffer Sprint? At least one per release
  - Milestone reviews?
    - Build / Increment / Iteration Review, TRR, RRD, RRT
    - Agile metrics at each sprint review (i.e., velocity)
      - Useful – burndown, burnup, velocity, features delivered, technical debt
  - Frequency of system-level demo? Monthly
  - Are you using Integrated Master Schedule (IMS)? Any alternative?
    - EVM at release/iteration level
    - Portfolio report (Jira)
    - SEER-SEM (agile)
    - Most productive is 80% assigned
    - PMI: 1 day is 6 hours

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Battle Rhythm

- **Where?**
  - **Default:** collocated team members
  - Challenges on distributed teams? Mitigations?
  - Do you have collocated users?
    - If not, how do you collaborate? How often?
      - Visit contractor site at appropriate times – open hot desk
      - Online access to Contractor dashboard
        - » Be careful on micromanagement
      - Skype, VTC helps but things lost in translation (facial, body language)
      - Need periodic person-to-person contact
  - Development environments? Demo environments? Staging or Operational-like environments?
    - System demo done in test or ops-like environment; depends on program
    - Leave development environment for development
    - Utilize Docker
    - Watch out for 'it works on my machine'

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Battle Rhythm

- **Why?**
  - **Default:** Four Manifesto Values and Twelve Principles
  - What works, what does not work?
    - Responding to change – sometimes means no change, expectation management, design for potential changes
    - For fixed price, Govt collaboration needs to be well understood
      - Be transparent, timely
    - Welcome changing requirements – typically no but tweak is ok

Individuals & interactions	over	**Processes** & tools
Working **software**	over	Comprehensive **documentation**
Customer **collaboration**	over	**Contract** negotiation
Responding to **change**	over	Following a **plan**

1. Satisfy the customer
2. Welcome changing requirements
3. Deliver working software frequently
4. Stakeholders work together daily throughout the project
5. Motivated individuals
6. Face-to-face conversation
7. Working software is the primary measure of progress.
8. Sustainable development
9. Continuous attention to technical excellence
10. Simplicity
11. Self-organizing teams
12. Continuous Improvement

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Battle Rhythm

- **How Many?**
  - **Default:** 4-9 people per team
    - No more than 10 members
  - Ratio between Product Owner and teams?
    - 1 Product Owner to 1-2 teams (max)
  - Ratio between Scrum Master and teams?
    - 1 Scrum Master to 1 team

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Agile Architecture / Architected Agile

Build "-ilities" and Resiliency in

- **Approaches:** Design-as-you-go, Emergent Design, Architecture Runway, Enterprise Architecture, Release Train
- What is your approach in developing architecture and design in Agile development?
  - Knowledge of Interfaces and Interoperability
  - Embedded Systems Engineering (including architect) team
  - Design in resilience; need team members educated on what resilience means
  - Architecture Runway – have SE/Design teams work ahead of development team to flush out design prior
- How do you address non-functional requirements?
  - Part of Definition of Done; every commit checks (mostly automated); peer review, performance testing
- How do manage dependency between components?
  - Roadmap needs to be clear with dependencies represented

GSAW 2018 Outbrief

Ground System Architectures Workshop

Session 11D

Pain Points (1/2)

- Buy-in at Middle Management
  - Use short term incentives; MVP for short term win
- Culture Shock
  - Leader is no longer the boss, acts as a facilitator
  - Transition: Processes, Metrics, Tools, Infrastructure, Role & Responsibilities
  - Hire an Agile Coach
  - Executive involved at the beginning
  - Government: increased workload tremendously
  - Contractor: matrix management of agile developers created risk across multiple projects
  - Agile is more costly in the beginning; cheaper in “total cost of ownership”

GSAW 2018 Outbrief

Ground System Architectures Workshop

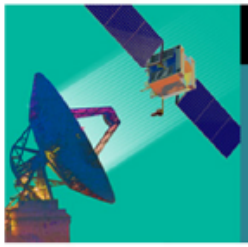
Session 11D

Pain Points (2/2)

- System Acceptance
  - Minimum at Feature and System level
  - Full System Test at the end
- Better way to adopt Agile, Process Improvement
- Midstream Agile Adoption
- RFP, FAR, Acquisition Milestones
- Scaling
- Interface to different processes
- System enhancement vs Greenfield development
- EVM, Project Planning, Quality Management
- Requirements Management

GSAW 2018 Outbrief

Ground System Architectures Workshop



Session 11D

Good idea

- Infrastructure and Resources must be ready (e.g., DevOps)
- Continuous Planning, Continuous Code Integration & Test
- Agile experience required; more important than certification
- Empower the team; but balance with checks and controls
- Release Engineer: Align integration team and system engineering team
- Assign 10% margin for reengineering
- **All parties need to be Agile**

Schedule

Time	Presentation and Discussion
1:00 – 1:20pm	Session Overview
1:20 – 1:45 pm	Agile Working Group 2018 Outbrief Jodene Sasine, The Aerospace Corporation
1:45 – 2:10pm	Scaled Agile in a traditional fixed contract world: A case from Satellite Monitoring and Control Enrique Fraga Moreira, GMV Aerospace and Defence
2:10 – 2:35pm	Revisit on Agile Fit Check Supannika Mobasser, The Aerospace Corporation
2:35 – 3:00pm	Agile Anti-Patterns Supannika Mobasser, The Aerospace Corporation
3:00 – 3:30pm	Break
3:30 – 5:00pm	General discussion • ***Smarter software factory and product delivery*** • ***Smarter program oversight and incentive structure*** • ***Smarter quality assurance, compliance, and accreditation*** • ***Smarter practices and other domains***

Schedule

Time	Presentation and Discussion
1:00 – 1:20pm	Session Overview
1:20 – 1:45 pm	Agile Working Group 2018 Outbrief Jodene Sasine, The Aerospace Corporation
1:45 – 2:10pm	Scaled Agile in a traditional fixed contract world: A case from Satellite Monitoring and Control Enrique Fraga Moreira, GMV Aerospace and Defence
2:10 – 2:35pm	Revisit on Agile Fit Check Supannika Mobasser, The Aerospace Corporation
2:35 – 3:00pm	Agile Anti-Patterns Supannika Mobasser, The Aerospace Corporation
3:00 – 3:30pm	Break
3:30 – 5:00pm	General discussion • ***Smarter software factory and product delivery*** • ***Smarter program oversight and incentive structure*** • ***Smarter quality assurance, compliance, and accreditation*** • ***Smarter practices and other domains***

Agile Fit Check - Revisit

Agile Fit Check - Revisit

Outline

- Agile Fit Check assessment
  - *Motivation*
  - *Literature Review*
  - *Agile Fit Check*
  - *Retrospective on common process selection criteria*
- Open discussion
  - *Improvements and impediments of agile adoption*
    - DevOps
    - Government Roles
    - User Involvement
    - Agile at the enterprise level

Agile Fit Check - Revisit

Agile software development in military domain

- It is a big challenge for large-scale government projects to follow a purely agile software development approach due to several constraints such as
  - *Barriers to customer-developer-user collaboration*
  - *Insufficient agile knowledge within the Program Office and its demands on the office*
  - *Scalability impacts on size, coordination, synchronization, criticality*
  - *Difficulty for Customers to speak as one voice, especially after series of requirements volatility*
- Several government “agile” programs adopt agile-like processes. Common characteristics are
  - *Time-box, frequent deliveries, continuous integration and testing*
  - *Iterative and incremental development*
  - *Very high collaboration, very high User involvement*

3

Agile Fit Check - Revisit

Motivation

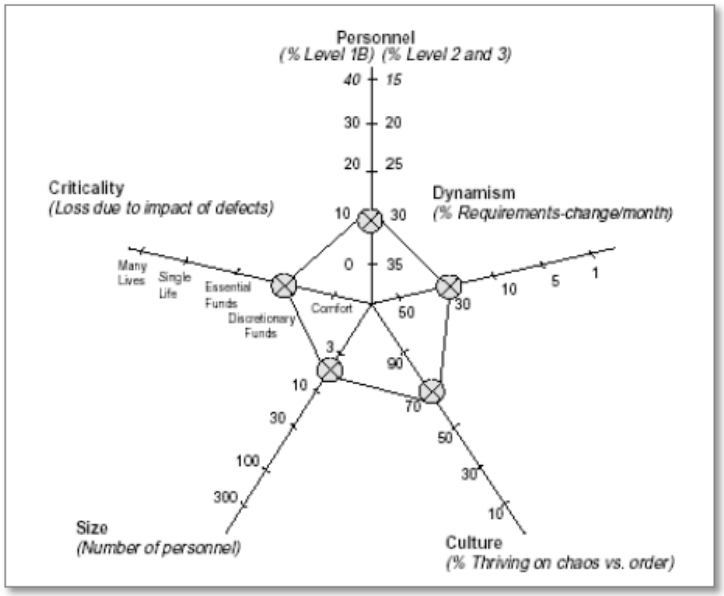
- There is growing trend of using an agile approach in system and software engineering projects in the industry
  - *Be agile, not do Agile*
- How do we know that a given project / a given proposal would be a good fit for an agile development?
- What are the evaluation criteria?
- What are the risks of adopting an incompatible process?

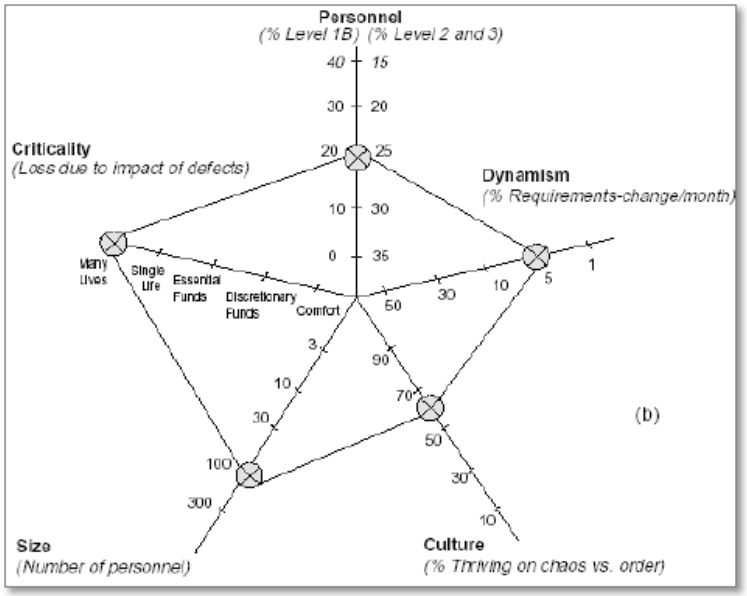
Agile Fit Check - Revisit

Boehm's Balancing Agility and Discipline

[Boehm and Turner 2003]

- Determine the relative suitability of agile or plan-driven methods
  - *Five critical decision factors*
  - *Scale ratings for each factor*
  - *Spider diagram to identify risk of agile adoption*
  - *Suggestions of risk mitigation*





Agile Fit Check - Revisit

MITRE - Traditional Versus Agile Considerations

[Modigliani and Chang 2014]

16 assessment areas to consider when adopting agile

Table 1 Traditional Versus Agile Considerations

Consider Agile Practices	Assessment Areas	Consider Traditional Practices
Requirements cannot be well defined upfront due to a dynamic operational environment.	Requirements Stability	Requirements have been relatively well defined by the operational sponsor.
Requirements can be decomposed into small tasks to support iterative development.	Requirements Divisibility	Requirements are tightly integrated and are difficult to decompose.
Users welcome iterative development and require frequent capability upgrades (<1 year).	User Timelines	Operational environment does not allow iterative development or lacks the ability to absorb frequent updates.
User representatives and end users are able to frequently engage throughout development.	User Involvement	Users cannot support frequent interaction with the development team or the target end user cannot be accessed.
Program scope is mostly limited to the application layer while using existing infrastructure.	Program Scope	Program spans core capabilities and underlying platform or infrastructure.
The government is responsible for primary systems integration.	Systems Integration	The government does not want to own systems integration responsibilities.
Capabilities are operational at a basic level, with some defects that can be addressed in future releases.	System Criticality	Program supports a critical mission in which defects may result in loss of life or high security risks.
Industry has relevant domain experience and Agile development expertise.	Developer Expertise	Agile development expertise is unavailable or lacks domain experience.
Program office has Agile training, experience, and/or coaches.	Government Expertise	Program office has no Agile experience or funding for Agile training or coaches.
Program contract strategy supports short Agile development timelines.	Contracting Timelines	Contract strategy cannot support short Agile development timelines.
Program Executive Office (PEO) or subordinate has authority for most program decisions.	Level of Oversight	Office of the Secretary of Defense (OSD) or Service Acquisition Executive (SAE) is the Milestone Decision Authority (MDA) and requires most decisions to be made at that level.
Development can be effectively managed by a small cross-functional government team.	Team Size	Many government stakeholders will be involved in the software development and decision-making process.
Government and developers can collaborate frequently and effectively.	Collaboration	Stakeholders physically located across multiple locations and have limited bandwidth to support frequent collaboration.
One or a few contractor(s) or teams can perform development.	Complexity	Many contractors are required to develop program elements.
Program can leverage test infrastructure and automated tests, and testers are active throughout development.	Test Environment	Extensive development and operational testing is conducted serially following development. Limited resources and tools available to conduct parallel development testing.
Leadership actively supports Agile development practices and provides "top cover" to use non-traditional processes and methods.	Leadership Support	Leadership prefers a traditional development approach or is unfamiliar with Agile practices.

6

Software Development Process Model Selection

[Welby 2013]

8 assessment areas to consider when adopting agile

Criteria for Software Development Process Model Selection

PMOs take on risk if processes don't fit their program.

Environment & Situation	Agile Indicator
Is the team for the work unit collocated or geographically dispersed?	Collocated
How many teams design, code & test, and integrate the effort?	Fewer is Better
Are you willing to rapidly change the system based on customer feedback?	Feedback more effective
Can incremental deliveries provide useful capability?	Easier to execute
Is there a fixed set of requirements against a fixed schedule?	Less backlog management benefit
Are all requirements the same priority? *(Closed Scope)*	Less prioritization benefit
Are change requests to requirements handled by the team or a separate organization?	Team management
Was the overall program schedule estimated assuming rolling-wave planning, or detailed bottom-up estimate?	Rolling Wave

Project Specific Decision: Reject One-Size-Fits-All Models

Agile Fit Check - Revisit

Agile Fit Check Framework

- Leveraged lessons learned from several agile-related publications, agile research studies, agile workshops / working groups, and agile programs
- Can be used to check the fitness of a proposal to use an agile development method on a program
- Does not give a yes/no answer on whether a program should use an agile development process
- Identifies potential impediments of adopting an agile approach for a particular system or risks of following an incompatible process

Agile Fit Check - Revisit

Agile Fit Check Criteria

- Checks for agile fitness based on criteria in three major categories:
  1. ***System's characteristics***
     - Q: Is the nature of the system applicable to agile development?
       - *Project scope, criticality, volatility, integration interval*
  2. ***Government's level of commitment***
     - Q: Is the Government ready to support an agile development?
       - *Leadership support, contract type, stakeholders' representatives*
  3. ***Contractor's level of commitment***
     - Q: Is the offeror or contractor ready to support an agile development?
       - *Collaboration, team organization*

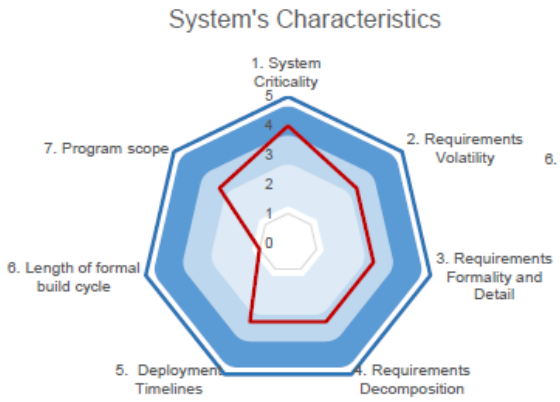
Agile Fit Check - Revisit

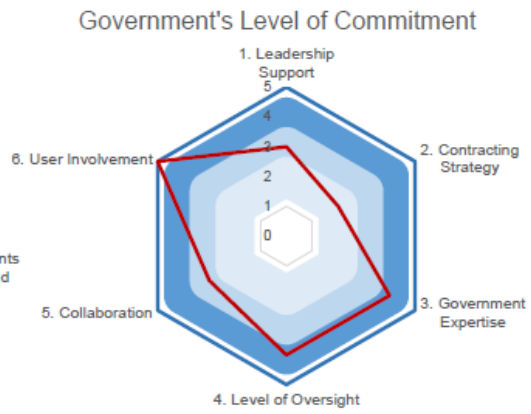
Agile Fit Check Criteria

[Mobasser 2017]

The following 19 factors can be used to determine the fitness of a certain project for an agile approach and can help identify the risks of adopting an incompatible process.

System's Characteristics	Government's Level Commitment	Contractor's Level Commitment
1. System Criticality	1. Leadership Support	1. Developer Expertise
2. Requirements Volatility	2. Contracting Strategy	2. No. of Contractor(s)
3. Requirements Formality and Detail	3. Government Expertise	3. Project Team Size
4. Requirements Decomposition	4. Level of Oversight	4. Supporting Infrastructure and environment
5. Deployment Timelines	5. Collaboration	5. Team Composition
6. System Integration Interval	6. End User Involvement	6. Use of automated testing
7. Program Scope		





9

Agile Fit Check - Revisit

Comparison of Process Selection Criteria

Mobasser 2017 (19)	Boehm and Turner 2003 (5)	GAO 2011 (9)	Welby 2013 (6)	Modigliani and Chang 2014 (16)	Hayes, et al 2016 (8)	Miller 2014 (33)
System Criticality	Criticality			System Criticality		
Requirements Volatility	Dynamism		Fixed requirements against schedule	Requirements Stability		Embrace requirements change
Requirements Formality and Detail						
Requirements Decomposition				Requirements Divisibility	Batch Size	
Deployment Timelines			Incremental deliveries			Interim delivery enabled
System Integration Interval					Synchronized Cadence	
Program Scope				Program Scope		
Leadership Support		Leadership support		Leadership support		Senior support for agile
Contracting Strategy						Appropriate contract type
Government Expertise	Personnel	staff knowledge and skills		Government Expertise		Appropriately trained staff
Level of Oversight		Program staff role and engagement		Level of Oversight	Product Owner Role	- Oversight-supported agile principles - Compatible rhythm of oversight
Collaboration		Regular communication		Collaboration		Collaboration enabled
User Involvement		User involvement		User Involvement	User Role	
Developer Expertise	Personnel			Developer Expertise		Agile/lean-competent staff
No. of Contractor(s)				Complexity		
Project Team Size	Size		Fewer teams	Team Size	Team Size	
Supporting Infrastructure and environment						Agile-supportive environment
Team Composition		Stable team				
Use of Automated Testing						
				User Timelines		
				Government as system integrator		
				Contracting Timelines		
				Test environment		- Positive change history - Trusting environment - Fail/learn fast
	Culture		Willing to rapid change			
					Specialization of Roles	
					Iteration Length	
					Release Definition	
			Collocated team			Co-located teams
			Requirements handled by the team			
			Rolling-wave planning			
			Prioritizable requirements			
		User participate in testing				
		SPO prioritized requirements				
		Sufficient funding				Project funding secured
						Clear program goals Defined success strategies Clear alignment of software and program goals Appropriate lifecycle activities Agile at-scale enabled Sponsors understand agile Cascading sponsorship External policy support Aligned incentives Agile-supportive reward system User and customer focus Review goals aligned with Agile Positive perception of Agile by team Appropriate use of cost-size factors Management as coaching function High trust between management and teams Sustainable development pace

Agile Fit Check - Revisit

Retrospective of Common Process Selection Criteria

Less of

- System Criticality
  - *Although still a concern, but more evidence of agile adoption in mission critical programs*
- Requirements Volatility / Stability
  - *Requirements stability is no longer an impediment to agile adoption*
- Project Team Size (not team size)
  - *Although still a concern, but more evidence of agile adoption in larger programs*
- Testing and Supporting Infrastructure and environment
  - *Major improvement on "Software Factory" and collaboration tools and environment*

Same of

- Leadership Support
- Developer Expertise
- Government Expertise
- Frequent System Level Integration and Incremental Deliveries
- Collaboration

More of

- User Involvement
  - *A Major challenge to get commitment from end user/operator for frequent feedback*
  - *Need three-star support*
- Product Owner Role / Level of Oversight
  - *No one-size-fits-all; need process tailoring*
- Contracting strategy
  - *Milestones, deliverables, incentives*
- Requirements Decomposition
  - *Horizontal component vs end-to-end feature*

Schedule

Time	Presentation and Discussion
1:00 – 1:20pm	Session Overview
1:20 – 1:45 pm	"Agile Working Group 2018 Outbrief" Jodene Sasine, The Aerospace Corporation
1:45 – 2:10pm	"Scaled Agile in a traditional fixed contract world: A case from Satellite Monitoring and Control" Enrique Fraga Moreira, GMV Aerospace and Defence
2:10 – 2:35pm	Revisit on Agile Fit Check Supannika Mobasser, The Aerospace Corporation
2:35 – 3:00pm	Agile Anti-Patterns Supannika Mobasser, The Aerospace Corporation
3:00 – 3:30pm	Break
3:30 – 5:00pm	General discussion • ***Smarter software factory and product delivery*** • ***Smarter program oversight and incentive structure*** • ***Smarter quality assurance, compliance, and accreditation*** • ***Smarter practices and other domains***

Agile Anti-Patterns

Supannika Mobasser, Brook Cavell,
Dan Ingold, Andrew Melnick,
Joanne Succari, Eric Yuan

Definition

Anti-Pattern:

"Antipatterns are common solutions to common problems where the solution is ineffective and may result in undesired consequences. An antipattern is different from bad practice when:

- *It is a common practice that initially looks like an appropriate solution by ends up having bad consequences that outweigh any benefits*
- *There's another solution that is known, repeatable, and effective.*
- *The concept of antipatterns was inspired by the concept of design patterns, which indicate common effective solutions to common problems.*

Antipatterns were initially applied in the context of software development, but have extended to other aspects of software engineering, organizations, and project management.

Coaches and consultants like to invoke antipatterns as a way of pointing out behavior they often see in teams they coach and as an avenue of suggesting better patterns."

Ref: https://www.agilealliance.org/glossary/antipattern

Examples of Anti-Patterns

Organizational

- **Analysis paralysis:** A project stalled in the analysis phase, unable to achieve support for any of the potential plans of approach
- **Groupthink:** A collective state where group members begin to (often unknowingly) think alike and reject differing viewpoints
- **Micromanagement:** Ineffectiveness from excessive observation, supervision, or other hands-on involvement from management
- **Mushroom management:** Keeping employees "in the dark and fed manure" (also "left to stew and finally canned")
- **Seagull management:** Management in which managers only interact with employees when a problem arises, when they "fly in, make a lot of noise, dump on everyone, do not solve the problem, then fly out"
- **Vendor lock-in:** Making a system excessively dependent on an externally supplied component

Project management

- **Cart before the horse:** Focusing too many resources on a stage of a project out of its sequence
- **Death march:** A project whose staff, while expecting it to fail, are compelled to continue, often with much overwork, by management which is in denial
- **Ninety-ninety rule:** Tendency to underestimate the amount of time to complete a project when it is "nearly done"
- **Overengineering:** Spending resources making a project more robust and complex than is needed
- **Scope creep:** Uncontrolled changes or continuous growth in a project's scope, or adding new features to the project after the original requirements have been drafted and accepted (also known as requirement creep and feature creep)
- **Brooks's law:** Adding more resources to a project to increase velocity, when the project is already slowed down by coordination overhead.

Ref: https://en.wikipedia.org/wiki/Anti-pattern#Examples

Anti-Patterns on New Agile Projects (1/3)

- **"Do Agile" vs "Being Agile"**
  - *Going through motions without understanding what intended outcomes should be*
    - Task 1 for requirements gathering, producing draft and final SRS IAW with IEEE std 830-1998; after three months, start Task 2, build product backlog, develop personas and user stories, sprint cadence etc.
    - SOW states the contract shall follow Agile methodology, shall define sprint cadence, etc. AND at kickoff, define the detailed capabilities and services to be delivered at the end of each sprint
    - Design review for each sprint
    - Death in CDRLs (25-40% overhead from IT CAST conference study)
    - Fluid Sprints – no time box or extended sprint length
    - Not planned well
    - Expectations that stories can move in and out or that sprint stays open if work is not completed
  - ***Recommendation***
    - Be Agile, follow Agile manifesto values, study Agile lessons learned

Anti-Patterns on New Agile Projects (2/3)

- **Scrum-but** : "we're doing Scrum, but we…" [do something that is completely the opposite of what it says to do in Scrum]
  - *Examples: Extensive up-front design, Large User Stories (all use cases covered), Lots of hand-offs (versus cross-functional team)*
- **Agile-on-the-fly:** If teams are new to Agile, it's recommended that they adopt it properly first, then try and experiment with it once they've got the hang of it.

- ***Recommendation***
  - *Need training, mindset change, full team (including management) buy-in, on-going coaching*

Anti-Patterns on New Agile Projects (3/3)

- **Assuming agile planning is entirely ad hoc and as-you-go**
  - *Agile planning is intended to be flexible, but not chaotic*
  - *One program planned "Releases," but had only the vaguest notion of what those Releases would contain*
    - Resulting in lack of structure and priority in their sprint planning
  - ***Recommendation***
    - Planning should be oriented to achieve a Minimum Viable Product or Minimum Operational Capability (focus on delivering features or capabilities rather than functional components) then enhance that capability incrementally in subsequent sprints and releases
    - NEVER be in a state where the product isn't working. Plan EVERY iteration to enhance and deliver additional mission capabilities.
- **Not building in quality**
  - *Not enough testing, especially regression testing, preferably automated regression testing*
  - ***Recommendation***
    - Start with the end in mind, use acceptance criteria, definition of done
    - Embed testers as part of the development team

Anti-Patterns on the Product Backlog Management or Requirements Management (1/3)

- Time/effort wasters
  - *Not spending time during backlog management - impacts Sprint Planning efficiency*
  - *Too many items or items too old – clutter and difficult to prioritize*
  - *Review/estimate everything (and too early) - unnecessary effort by team*
  - *Too much information or acceptance criteria – leave room for discussion with the team for new perspectives and negotiation on scope; leaves team less engaged if everything is spelled out*
  - ***Recommendation***
    - Organize backlog grooming sessions to ensure that items are ready for next Sprint
    - Review/estimate the top priority items that are likely to be addressed in the next 1-2 sprints

Anti-Patterns on the Product Backlog Management or Requirements Management (2/3)

- Prioritization issues
  - *Prioritization by proxy – someone else (external to Product Owner) dictates the priorities; no accountability of Product Owner*
  - *Prioritize full project up front*
  - ***Recommendation***
    - Appoint a Product Owner with authority
    - Priorities should expect to adjust as you observe working software and assess how much is enough

Anti-Patterns on the Product Backlog Management or Requirements Management (3/3)

- Failing to organize and prioritize the backlog by feature
  - *Item is horizontal component versus end-to-end feature*
    - May lead to completed components, but not operational or not delivering mission value
- Composition of a backlog item
  - *Items not decomposed from Themes or Epics*
- Assign a non-functional requirement to be developed in one Sprint
  - *E.g. scalability requirement can not be achieved in one Sprint*

- ***Recommendation***
  - *Factor in or bake in non-functional requirements from day one*
  - *Start from requirements decomposition. Organize requirements in a capability-driven structure*
  - *Helpful to be able to filter by an epic and see what the features have been defined and help to convey scope of a particular release*

Anti-Patterns on Architecture and Design

- Architecture and Design are somewhat orthogonal to development methodologies
  - *Agile doesn't work well with stovepipes or monolithic hardware/software systems*
  - ***Recommendation***
    - Modularity, layered architecture, abstracted dependencies
    - Small, self-contained, testable feature decomposition
    - Start with Just Enough Architecture
- Several space and ground software are developed by engineers from other domains without software engineering background
  - ***Recommendation***
    - Be cognizant of technical debt
    - Start by defining some principles, tenets and architectural decisions upfront
- Lack of Government-owned software architecture
  - ***Recommendation***
    - Define interface specifications

Anti-Patterns on Testing

- Accepting manual testing as suitable and effective
  - ***Recommendation***
    - Design for test
    - Use Test-Driven Development at the unit level
    - Use Behavior-Driven Development at the integration level
    - Continue to enhance automated functional tests throughout the lifecycle
- Accepting automated unit testing covers everything
  - ***Recommendation***
    - Need a good balance between automated and manual testing
- Not integrating Test with Development
  - ***Recommendation***
    - Test as you go, continual testing, automated regression testing

Anti-Patterns on the Software Development Team

- Product Backlog Grooming done privately
  - ***Recommendation***
    - Full team must be involved to ensure a shared understanding of the “why” and “what” since anyone on the team should be able to pick up a story and work on it
- Fill Sprint Backlog with 100% of the team capacity
  - *Lead to “I’m busy” attitude and no time to help others*
  - ***Recommendation***
    - Leave room for collaboration and team support
- Recommend to identify a “Story Shepherd” who ensures the tasks and task dependencies are clear and moving along within the sprint

Anti-Pattern on Project Management (1/2)

- Planning failures
  - *Pre-planning iterations as though using traditional planning (IMS)*
  - *Failing to plan releases in terms of expected features/capabilities*
  - *Creating a too-detailed IMS, and evaluating progress (and EV) against it*
  - ***Recommendation***
    - May continue to use high-level IMS
    - Use more of Product Roadmap and Architecture Runway
      - *Consider dependencies and constraints when developing Release Plan*

- Team organization failures
  - *Teams organized by functional decomposition, rather than by Feature*
  - ***Recommendation***
    - Encourage multi-disciplinary teams organized by Feature/Epic
    - Focus on flat organization structure to speed up decision-making process

Anti-Pattern on Project Management (2/2)

- Tasks too large that it sits in “In Progress” the whole sprint
  - *“Yesterday I worked on X and today I plan to work on X”*
    - Team doesn’t really know what you are doing or whether progress is being made
    - It’s not apparent if you are blocked or stuck on a problem that possibly someone else can help with
    - Impacts dependent subtasks and likely the larger story
  - ***Recommendation***
    - Scrum master should pay attention to progress, shared vision, and potential impediments
- Ineffective retrospective
  - *No action or follow-up taken to remedy issues*
    - Repetition of issues, deteriorating morale
  - *Team less likely to raise concerns if they feel nothing will change*
  - ***Recommendation***
    - Team should identify actionable, measurable, and controllable items
    - Put action items in the backlog
    - Review past action items with the team

Schedule

Time	Presentation and Discussion
1:00 – 1:20pm	Session Overview
1:20 – 1:45 pm	Agile Working Group 2018 Outbrief Jodene Sasine, The Aerospace Corporation
1:45 – 2:10pm	Scaled Agile in a traditional fixed contract world: A case from Satellite Monitoring and Control Enrique Fraga Moreira, GMV Aerospace and Defence
2:10 – 2:35pm	Revisit on Agile Fit Check Supannika Mobasser, The Aerospace Corporation
2:35 – 3:00pm	Agile Anti-Patterns Supannika Mobasser, The Aerospace Corporation
3:00 – 3:30pm	Break
3:30 – 5:00pm	General discussion • ***Smarter software factory and product delivery*** • ***Smarter program oversight and incentive structure*** • ***Smarter quality assurance, compliance, and accreditation*** • ***Smarter practices and other domains***

Smarter Software Factory

- Do you agree with the following minimum essential elements of a software factory?
  - *Continuous integration*
  - *Continuous testing*
  - *Tool chain with maximum automation*
  - *Reusable code*

- How can we make it smarter?
  - *Templates : Pre-made application elements with placeholders for arguments.*
  - *Recipe : Automate procedures in routine tasks*
  - *Architecture guidance and patterns*
  - *IV&V with machine learning?*
  - *Data-driven*
  - *Cloud-based?*
  - *Continuous deployment*
    - Should we / can we do that? Deploy to where?

Smarter Cybersecurity Compliance

- Do you agree with the following minimum essential elements of Cybersecurity approach?
  - *Automated Testing/Test Reporting*
  - *Automated Security Scanning*
  - *CI/CD integrated with source code scans (security and quality)*
  - *All deployment candidates scanned prior to deployment*

- How can we make it smarter?
  - *Automated compliance monitoring*

Smarter Certification and Accreditation

- Do you agree with the following minimum essential elements of certification and accreditation process?
  - *Plan for early and upfront involvement*
  - *Define as part of acceptance criteria and definition of done*

- How can we make it smarter?
  - *Composable certification [DARPA 2018]*
    - Use the evaluated criteria of a subsystem as evidence in a system evaluation
  - *Automated evaluation [DARPA 2018]*
    - Produce compelling, checkable assurance arguments backed by evidence
  - *Data-driven evidence*

Smarter Government-Led Testing

- How can the government test be performed early and often?
  - *How early?*
  - *How often?*
    - Sprint-level, quarterly, annually, one-time

- How can we make it smarter?

Smarter Product Delivery

- Do you agree with the following minimum essential elements of product delivery?
  - *Evolutionary and incremental delivery of capability*
  - *Design, develop, and plan for continuous re-hosting*
  - *Produce the software in the operational environment*

- How can we make it smarter?

Smarter Program Oversight

- Do you agree with the following minimum essential elements of mission assurance?
  - *Govt has online access to contractor's real-time repository or development environment*
    - Need to balance with micro management
  - *Real-time dashboard*
  - *Wiki-based documentation*
  - *As-built & incremental deliverables*
  - *Govt participates in Sprint/Iteration Reviews*
  - *Frequent system-level integration (at minimum monthly)*

- How can we make it smarter?

Smarter Incentive structure

- “Be careful what you wish for”

- From Govt to contractor
  - *What to incentivize?*
    - Specific goal? Stretch goal? Innovation? Schedule? Quality?
  - *What not to incentivize?*
- From high level management to development team
  - *What to incentivize?*
    - Specific goal? Stretch goal? Innovation? Schedule? Quality?
  - *What not to incentivize?*

Smarter Government Involvement

- What would be a smarter approach for the Government teams to be involved?
  - *Government program management*
  - *Operators / Users*
  - *Government sustainment team*
  - *Certifier / Appraiser*

Smarter Agile and MBSE

- MBSE – Model-based Systems and Software Engineering
  - *Such as requirements, diagrams, simulations, prototype*
  - *“Model-first, code-later” vs “lo-fi from developers, then hi-fi by modelers”*

- How can we make it smarter?

Smarter Agile and Hardware-intensive development

- Challenges
  - *Do you need to complete the requirements and design before coding?*
  - *What do the milestones or synchronization points look like?*
  - *Simulation-in-the-loop*

- How can we make it smarter?

Reference

[Boehm and Turner 2003] Barry Boehm and Richard Turner, "Balancing Agility and Disciplines", Addison Wesley 2003

[DARPA 2018] Raymond Richards, "Technical challenges in certifying software for military systems", SERC workshop: Continuous Development and Deployment of Military Capabilities, November 27-28, 2018

[GAO 2011] GAO, "Critical Factors Underlying Successful Major Acquisitions", GAO Report-12-7, October 2011

[Hayes, et.al 2016] William Hayes, Mary Ann Lapham, Suzanne Garcia-Miller, Eileen Wrubel, and Peter Capell, "Scaling Agile Methods for Department of Defense Programs", CMU/SEI-2016-TN-005, December 2016

[Miller 2014] Suzanne Miller, "The Readiness & Fit Analysis: Is Your Organization Ready for Agile?", CMU/SEI white paper 2014-019-001-90981, April 2014

[Mobasser 2017] Supannika Mobasser, "Agile Fit Check Framework for Government Acquisition Programs", CSER 2017

[Mobasser 2018-2] Supannika Mobasser, "Agile Fit Check: Is your program fit for Agile", SERC workshop: Continuous Development and Deployment of Military Capabilities, November 27-28, 2018

[Mobasser and Sasine 2018] Supannika Mobasser and Jodene Sasine, "GSAW 2018 Working Group Outbrief - Achieving Resiliency with Agile Methods", GSAW, February 26 – March 1, 2018

[Modigliani and Chang 2014] Peter Modigliani and Su Chang, "Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities", MITRE. March 2014

[VersioneOne 2018] The 12th Annual State of Agile Report, VersionOne, April 2018

[Welby 2013] Stephen Welby, "Thinking About Agile in DoD", AFEI Agile for Government Summit, November 2013

Back up charts

Manifesto for Agile Software Development

http://www.agilemanifesto.org/

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

That is, while there is value in the items on the right, we value the items on the left more."

Individuals & interactions	over	**Processes** & tools
Working **software**	over	Comprehensive **documentation**
Customer **collaboration**	over	**Contract** negotiation
Responding to **change**	over	Following a **plan**

[Ref: Agile manifesto http://www.agilemanifesto.org/]

Agile development promotes

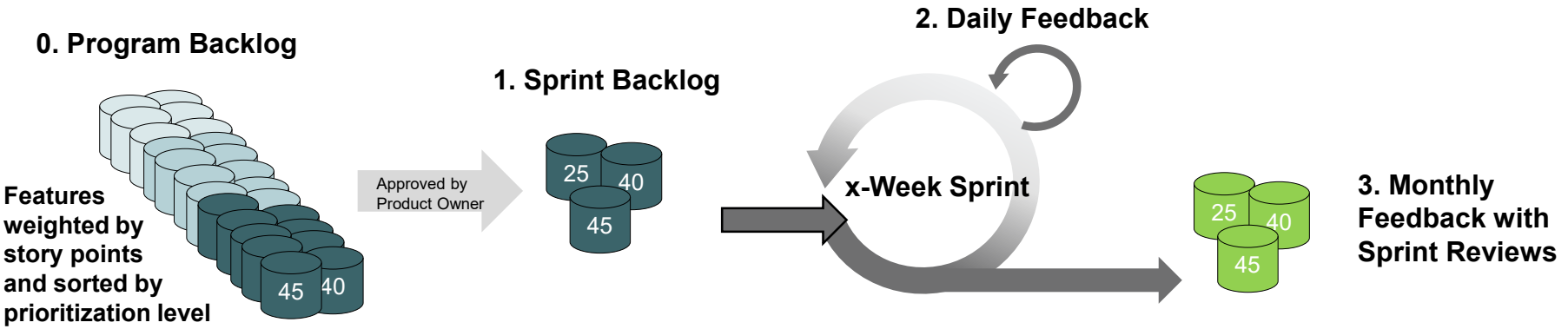
- Adaptive planning
- Evolutionary development and delivery
- Time-boxed iterative approach
- Rapid and flexible response to change

12 principles of Agile software development

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence** and good design enhances agility.
10. **Simplicity-** the art of maximizing the amount of work not done--is essential
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the team **reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Ref: https://agilemanifesto.org/principles.html

Agile Methodologies – Scrum



Four-Week Sprints (Time-Boxed) Used to Design, Develop, Integrate, & Test Selected Software Features

0. Requirements are used to create a **program backlog,** a prioritized list of software features.

Each **feature** gets a relative difficulty/time rating in **story points.** Each **feature** is assigned its **priority** level.

The Government team approves the size and priority of each feature.

1. **Sprint Backlog** for each monthly sprint, developers commit to delivering a set of features captured in a sprint backlog.

The Government team, represented by the **Product Owner**, approves the selected **sprint backlog.**

2. Daily feedback:
a. Teams get status & problem alerts via daily 10-15 minute stand-up.

b. Continuous integration and Automated testing of code means that code is checked in, built, and regression tested at least once every day

c. The Government Team has access to up-to-minute, web-based metrics, provide quick feedback

3. Monthly feedback with Sprint Review for both development team and the Government team.

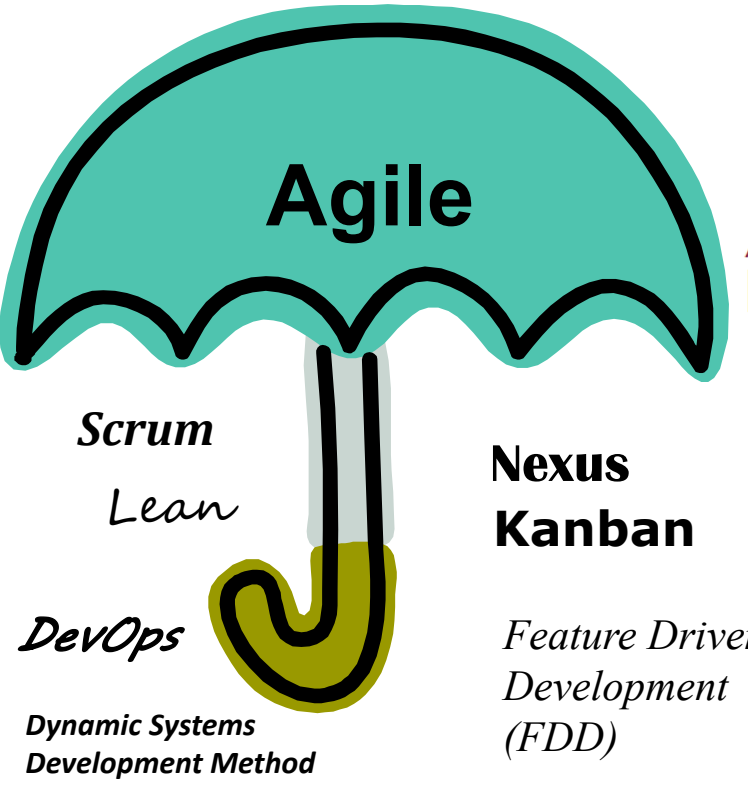
Feedback on planning accuracy and progress-to-date. Features aren't counted as **Done** until they are integrated & tested successfully.

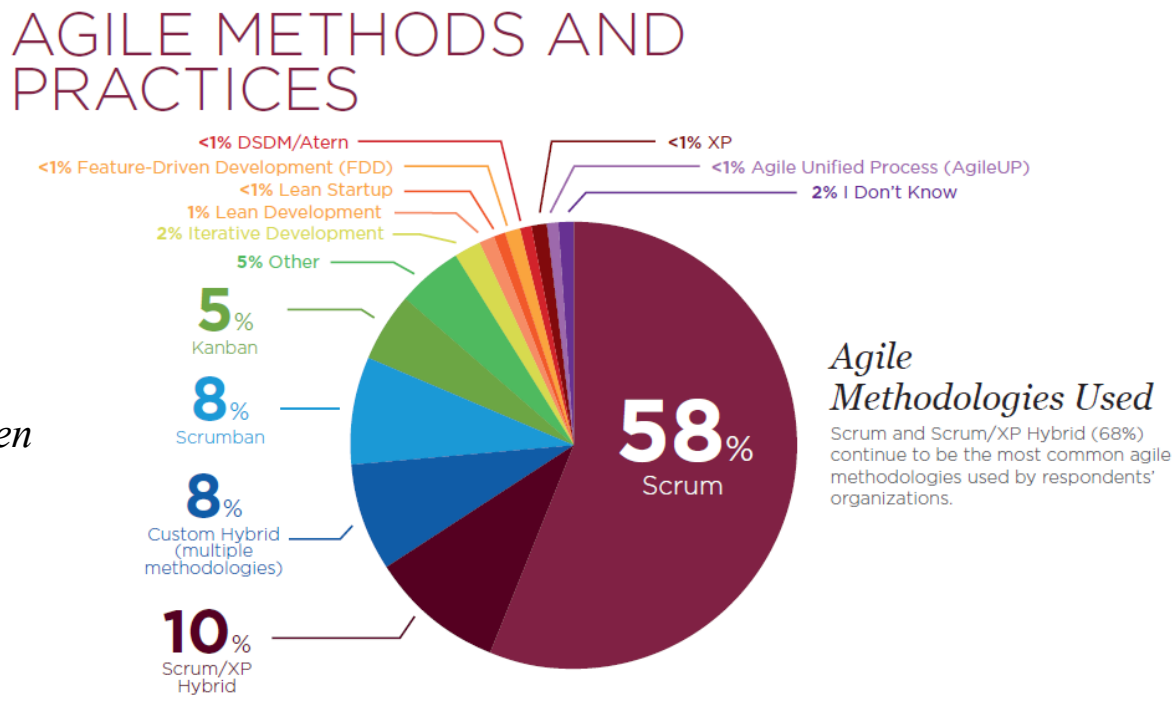
Acceptance Testing.

The development team performs **Sprint retrospective.**

Agile Methodologies

Scrum: the most popular Agile methodology in the commercial sector





[State of Agile Survey Report, VersionOne, 2018]