Agile Software Development
Agile Terms

- Sprint
- Velocity
- User Story
- Burndown
- Backlog
- Backlog Grooming
- Definition of “Done”
- Scrum Master
- Planning Poker
- Iterative
- Retrospective
- WIP
- Kanban
- Scrum
- Empirical
- Agile Manifesto
- Product Owner
- Story Points
- Daily Standup
- Incremental
Today we will discuss:

• Why should I consider using an Agile Software Development Process?
• Introduction to Agile Software Development Processes
• Closer Look at Scrum
• Metrics
• Summary
Software provides the decisive advantage to the warfighter
  - Critical to future battlefield dominance

Software Acquisition for major programs poses some of our toughest System Engineering Challenges
  - Compounding the challenge: the only constant for DoD systems is change
    - Evolving Threats
    - Rapid Technological Change
    - Strategic and Tactical Innovation

These factors all demand increased agility for military systems
  - Designs that afford capacity to adapt and adjust
  - Maintaining operational advantage in an environment of change
  - Ever Increasing functionality controlled by software: Software can enable system change

Software Development Agility: Key Contributor to Program Success

Reference: 18th NDIA SE Conference, Oct 2015, Mr. Sean Brady, Office of the Deputy Assistant Secretary of Defense for Systems Engineering
**Agile projects are 3 times more successful than Waterfall**

**Successful** = delivered on time, on budget, with all required features and functions  
**Challenged** = Late, over budget, and/or with less than required features and functions  
**Failed** = Cancelled prior to completion or delivered and never used  

• DoD Leaders increasingly appreciate Agile Software Practices and Growing Adoption
• DoD seeks practical approaches that mesh Agile with DoD’s statutory, regulatory, operational, and closed-scope environment
• DoD applauds any methodology that can improve Software Acquisition Outcomes
• DoD 5000.02 supports tailoring for adoption of Agile Software Development

Reference: 18th NDIA SE Conference, Oct 2015, Mr. Sean Brady, Office of the Deputy Assistant Secretary of Defense for Systems Engineering
• Statute: 10 U.S.C. 2366b – MDAP certification required before Milestone B
  - Approval requires PDR and formal post-PDR assessment
  - Certifies program “demonstrates a high likelihood of accomplishing its intended mission”
• Regulation: DoDI 5000.02 – PDR/CDR Assessments
  - MDAPs/MAIS: a post-PDR and -CDR assessment to the MDA
  - Assess design maturity, and the program’s readiness to begin software coding with acceptable risk
• DAG Guidance for PDR:
  - … software architecture designs have been established; all Computer Software Units (CSUs) have been defined.
  - Software Requirements Specification (SRSs) and Interface Requirement Specifications (IRRs), are complete for all CSCs
• How does DASD(SE) formally assess post-PDR / CDR assessment given
  - Evolving requirements and design maturity
  - Evidence of enough SE Rigor to ensure “high likelihood of program accomplishing its intended mission?”
  - … when we have allocated baseline for a small fraction of the system?
  - Acceptable tailoring decisions including minimums related specifically to:
    - PDR/CDR, related documentation / artifacts
    - Allocated and product baseline content, and delivery points?

Reference: 18th NDIA SE Conference, Oct 2015, Mr. Sean Brady, Office of the Deputy Assistant Secretary of Defense for Systems Engineering
Increased Probability of Success!
• GAO Performed a study in 2012 titled “Effective Practices and Federal Challenges in Applying Agile Methods”
  - To reduce the risk of such problems, the Office of Management and Budget (OMB) recommends modular software delivery consistent with an approach known as Agile
• Government produced the TechFar “Handbook for Procuring Digital Services Using Agile Processes”
  - “In the Government, digital services projects too often fail to meet user expectations or contain unused or unusable features. Several factors contribute to these outcomes, including the use of outdated development practices and, in some cases, overly narrow interpretations of what is allowed by acquisition regulations."
“Simply delivering what was initially required on cost and schedule can lead to failure in achieving our evolving national security mission — the reason defense acquisition exists in the first place.”

Honorable Frank Kendall Under Secretary of Defense (AT&L) 2015 Performance of The Defense Acquisition System

DoD must be responsive to change.
Today we will discuss:

• Why should I consider using an Agile Software Development Process?
• Introduction to Agile Software Development Processes
• Closer Look at Scrum
• Metrics
• Summary
Agile software development is a set of software development methods in which requirements and solutions evolve through collaboration between self-organizing,[1] cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.[2]


Agile Manifesto

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

- Individuals and Interactions  Over  Processes and Tools
- Working Software  Over  Comprehensive Documentation
- Customer Collaboration  Over  Contract Negotiation
- Responding to Change  Over  Following a Plan

That is, while there is value in the items on the right, we value the items on the left more.
Agile Software Development Umbrella

- Scrum
- Extreme Programming (XP)
- Lean
- Kanban
- Crystal
- Feature-Driven Development (FDD)
- Dynamic Systems Development Methodology (DSDM)
The empirical model of process control provides and exercises control through frequent inspection and adaptation for processes that are imperfectly defined and generate unpredictable and unrepeatable outputs.
Today we will discuss:

- Why should I consider using an Agile Software Development Process?
- Introduction to Agile Software Development Processes
- Closer Look at Scrum
- Metrics
- Summary
Scrum is an iterative and incremental software process framework

- Relies on collaborative, self-organizing, cross-functional, self-managing teams
- Empirical approach to managing the process
- Responsive to emerging or changing requirements
- Focuses on delivering value quickly
• Time-boxed iteration of software development
  - Typically 1 – 4 weeks in duration
  - Should not exceed a calendar month
• Produces an increment of potentially shippable software
• Once Team commits to the sprint they are not interrupted with any changes that would affect them reaching their sprint goal.
Scrum Framework Overview

Initial Planning
- Capture User Stories
- Release Planning
- Product Backlog

Sprint
- Start of Sprint
- Daily Scrum
- Implementation
- End of Sprint
- Sprint Review
- Sprint Retrospective

Scrum Key
- Events
- Artifacts
- Measures
- Other

“Done” Software
Velocity

Sprint Backlog
Burndown
Scrum Roles

**Product Owner (The What)**
- Owns the Product Vision
- Responsible for maximizing the value of the product
- Manages the Product Backlog
- Sets the priorities for items in the backlog
- Ensures development team has clear understanding of the backlog items being worked
- Grooms the backlog periodically to ensure priorities are correct, items are appropriate level of detail, and newly identified requirements are captured

**Development Team (The How)**
- Owns the Execution and Delivery
- All the team members who work to produce the “potentially” shippable increment of software
- Self-Organizing
- Cross-Functional
- Small group: 3-9 Members
- Manages own work during the sprint

**ScrumMaster (The Coach)**
- Owns the Process Vision
- Removes impediments that are preventing development team from completing work
- Ensures that the Development Team understands and is following the process
- Helps the Product Owner understand the process and their role
- Looks for ways to improve productivity
**Scrum Roles**

- How does the distribution of role responsibilities affect the authority vested in a given role?

<table>
<thead>
<tr>
<th>Scrum Master</th>
<th>Product Owner</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sponsor</td>
<td></td>
</tr>
<tr>
<td>PM/APM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APM-E</td>
<td>Cyber</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Log</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Developer</td>
<td></td>
</tr>
</tbody>
</table>
## Scrum Artifacts

### Product Backlog
- Prioritized list of user stories, features, or work items detailing the desired functionality for the product being developed.
- Items in the backlog are often referred to as Product Backlog Items (PBIs).
- Managed by the Product Owner.
- Acronym DEEP used to illustrate attributes of a good backlog:
  - D – Detailed Appropriately
  - E – Estimated
  - E – Emergent
  - P – Prioritized

### Sprint Backlog
- Is the plan to get to “Done” for the given Sprint.
- Created during Sprint Planning.
- Consists of the PBIs committed for the Sprint, plus all tasks necessary to get those PBIs to the done state during the sprint.
- Usually the tasks contain hours remaining estimate to enable daily tracking of progress. (Used to create Burndown Chart).
- Owned by the Development Team.
- Highly visible real-time picture of the work being done.

### Done Software
- “Done” means the software is potentially shippable.
- Team creates the “Definition of Done”.
- “Definition of Done” is a checklist owned by the team that list all activities that must be completed before a PBI will be considered done.
- Example “Definition of Done”:
  - Passes Acceptance Criteria
  - Code Peer Reviewed
  - All unit tests pass
  - Builds without warnings
User Story

• Written
  - “As a <type of user>, I want <some goal> so that <some reason>”

• Conversations
  - Flesh out the details

• Acceptance Criteria
  - Defines what constitutes when the story is complete

• Epic
  - A large user story that will usually be broken up before being implemented
Example User Stories

- As a maintenance test pilot, I want to be able to view all the readings for a selected flight data parameter graphed over time so that I can review what occurred during the flight.
- Epic – As a user, I can backup my hard drive
  - As a power user, I can specify files or folders to backup based on file size, date created and date modified
  - As a user, I can indicate folders not to backup so that my backup drive isn’t filled up with things I don’t need to save [1] Reference Mike Cohn: Mountain Goat Software

Test:
- Short, simple, customer-centric statement
- It describes a discrete product function that is required
- It focuses on describing what will provide customer value
- Usually recorded on physical cards (Kanban Board Use)
• **EPIC/Capability:** A large grained definition of a need. One or more releases are required.

• **Feature:** A discrete and coherent component of an EPIC/Capability. Planned to complete within a Release

• **User Story/Backlog Item:** Part of a Feature that can be estimated in size and complexity and prioritized in Sprint backlog. Planned to complete within a sprint.

• **Tasks:** Planned to complete within 2 to 8 hours. Tasks are estimated by hours.
Sprint Planning Meeting

• Who
  - Team, ScrumMaster, and Product Owner

• Agenda
  - Discuss the top priority product backlog items
  - Team selects which to do

• Why
  - Know what will be worked on
  - Understand it enough to do it

“A good plan violently executed now is better than a perfect plan executed next week.” — General George S. Patton
Estimation: Story Points

- Unit of measure for evaluating the relative size of a user story, feature, or work item
- Typical Estimation Scale roughly follows the Fibonacci sequence
  - 0, 1, 2, 3, 5, 8, 13, 20, 40, 100
  - Some teams include $\frac{1}{2}$ and $\infty$
- A story assigned a two should be roughly twice the size of a story assigned a value of one
- Unit-less Measurement
- Delphi / Expert Judgement method for estimating
Daily Scrum

- Development team meets to make the plan for the day
- Answer 3 questions:
  - What did I do yesterday?
  - What am I doing today?
  - What are my impediments
- Time Boxed
  - 15 Minutes, standup
- Scheduled daily for the same time and place
- Opportunity to inspect and adapt daily based on the current progress on completing the sprint backlog

These are NOT status for the ScrumMaster. They are commitments in front of peers
Sprint Review

• Demonstrate the work completed during the sprint
• Opportunity to Inspect the work product and adapt the plan if needed
• Scrum Team and Stakeholders collaborate about what was completed in the sprint
• Opportunity to make changes to Product Backlog based on feedback
• Chance to discuss the path forward for the next sprint
Sprint Retrospective

• Review the Process and Product looking for ways to improve either
• Answer Three Questions
  - What worked?
  - What didn’t work?
  - What will we do differently?
• Continuous Process Improvement
Scrum Summary – Key Components of the Scrum Framework

- **4 Activities**
  - Sprint Planning
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

- **3 Roles**
  - Product Owner
  - Development Team
  - ScrumMaster

- **3 Artifacts**
  - Product Backlog
  - Sprint Backlog
  - “Done” Software
Integration with SETR

**Stage A**
- **Solution Engineering**
  - Explore the program solution to ensure all alternatives are considered

**Stage 1**
- Planning
  - Plan the project and acquire resources needed to achieve solution

**Stage 2**
- Requirements Definition
  - Analyze user needs and document functional requirements

**Stage 3**
- Design
  - Transform requirements into detailed system design

**Stage 4**
- Development
  - Convert the system design into system

**Stage 5**
- Integration and Test
  - Integrate with other systems: develop C&A

**Stage 6**
- Implementation
  - System moved to Production environment: Production data has been loaded

**Stage 7**
- Operations and Maintenance
  - The system in operated to carry out intended function

**Stage 8**
- Disposition
  - The system is disposed

**Key**
- **SPR:** Study Plan Review
- **SER:** Solution Engineering Review
- **PPR:** Project Planning Review
- **SDR:** System Definition Review
- **PDR:** Preliminary Design Review
- **CDR:** Critical Design Review
- **IRR:** Integration Readiness Review
- **PRR:** Production Readiness Review
- **OTRR:** Operational Test Readiness Review
- **ORR:** Operational Readiness Review
- **PIR:** Post Implementation Review

*APC1383_028a*
Integration with SETR
Today we will discuss:

• Why should I consider using an Agile Software Development Process?
• Introduction to Agile Software Development Processes
• Closer Look at Scrum
• Metrics
• Summary
Software metrics are used to obtain objective, reproducible measurements that can be useful for quality assurance, performance, debugging, management, and estimating costs.

**Product**
- Size (SLOCs, function points), performance, complexity, quality attributes (reliability, usability, etc.)

**Process**
- Defect removal efficiency, programmer productivity, etc.

**Project (Management)**
- Cost, schedule, EVM, trouble report statistics, errors per man-month, etc.

**Use metrics to:**
- Define a minimal set
- Tailor it to management needs
- Collect across the lifecycle
- Manage Risk
- Track technical progress
- Improve process
- Evaluate SW practices
- Use automated collection
## Traditional Metrics

<table>
<thead>
<tr>
<th>Issue</th>
<th>Example metrics/measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule and Progress</td>
<td>Milestone dates, Requirements Status/Growth, Problem Report Status, Critical Path Performance, Planned vs. Actual Units Designed, Coded, Unit Tested, and Integrated.</td>
</tr>
<tr>
<td>Resources and Cost</td>
<td>Effort, Staff Turnover, Earned Value (CV, SV), Planned vs. Actual Staffing Levels.</td>
</tr>
<tr>
<td>Product Size and Stability</td>
<td>Source Lines of Code (SLOCs), Function Points, Number of Interfaces, Database Size.</td>
</tr>
<tr>
<td>Process Performance</td>
<td>Reference Model Rating, Programmer Productivity, Cycle Time, Defect Rate, Defect Removal Efficiency.</td>
</tr>
<tr>
<td>Technology Effectiveness</td>
<td>Requirements coverage.</td>
</tr>
<tr>
<td>Customer Satisfaction</td>
<td>Survey Results, Requests for Support</td>
</tr>
</tbody>
</table>
Integrating Agile Measurement into Your Overall Management Metrics

Agile/Iterative measures for teams leverage the approach of time-boxing (fixing the schedule for the iteration, and varying the target requirements to be fulfilled based on the team’s capacity and capability)

- Opposite of traditional acquisition where schedule is based on estimates of what it takes to implement a fixed set of requirements

Agile/Iterative iteration measures are meant primarily for the team’s use

- External use of iteration metrics is generally discouraged
- Release metrics are generally used for monitoring/management purposes
Teams focus on delivering working code.

Typical Team Measures for Agile Development

Metrics used by and for the development team:

- Kanban Board for Task Tracking
- Sprint Burn-down Charts
- Release Burn-Up Charts
- Velocity
Burndown Chart

- Tool for visualizing daily progress during a sprint
- Ideal trend line graphs hours remaining for tasks in sprint backlog over the length of the Sprint
- As team updates hours remaining for the tasks in progress the chart indicates if progress is above or below the trend line
Velocity

• Velocity – Is a measure of a team’s rate of progress
• The purpose of velocity is to help aid the team in planning the next sprint and forecasting when to expect a subset of features to be ready for shipping
• Calculated by summing the story point estimates of the PBIs completed during the Sprint
• The average velocity is used to determine how much to plan for the next sprint
• **NOTE:** Velocity is tied to a specific team and should not be used to compare two teams!
**Cumulative Flow Diagrams**

- The green line/area shows how many items have been delivered over time.
- The yellow line / area shows how many items are in progress.
- The top part (red) is the backlog or how many items weren’t yet started.

**Formulae:**
- $X = \text{WIP Queue Duration (Lead Time)}$
- $Y = \text{WIP Queue Length (Number Items in Progress)}$
Today we will discuss:

• Why should I consider using an Agile Software Development Process?
• Introduction to Agile Software Development Processes
• Closer Look at Scrum
• Metrics
• Summary
Agile is not a Silver Bullet, But DoD can benefit greatly from adopting Agile processes
- Agile can reduce program risk by frequent releases
- Agile can respond to change to ensure that the warfighter is actually getting a product they need
- Agile allows the DoD to integrate the latest technological advancements to address an increasingly complex operational environment to allow them to keep pace with the rapid changes in IT

Agile provides a core set of principles to help the DoD get away from cumbersome monolithic systems for IT
- DAU – CLE 076 (Coming Soon)
- DAU Agile Workshop
- GAO 12-681 Report – Federal Challenges and Best practices with Agile
- TechFAR from 18F - Handbook for Procuring Digital Services Using Agile Processes
- PARCA Agile and EVM PM Desk Guide
Backup Slides
• Each team member has a set of valid Story Point Cards
• Each Product Backlog Item (PBI) is discussed briefly, reference story chosen
• Discuss each story (2-3 mins)
• Each Development Team Member selects a card matching their estimate
• Everybody shows their cards at same time
• Discuss Outliers (high/lows)
• Repeat until team reaches a convergence in estimates
• Record estimate for the PBI

<table>
<thead>
<tr>
<th>Card(s)</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Task is already completed</td>
</tr>
<tr>
<td>½</td>
<td>The task is tiny</td>
</tr>
<tr>
<td>1, 2, 3</td>
<td>Small tasks</td>
</tr>
<tr>
<td>5 &amp; 13</td>
<td>Medium sized tasks</td>
</tr>
<tr>
<td>20 &amp; 40</td>
<td>Large sized tasks</td>
</tr>
<tr>
<td>100</td>
<td>Very large tasks</td>
</tr>
<tr>
<td>∞</td>
<td>Huge task</td>
</tr>
<tr>
<td>?</td>
<td>I have no idea how long this task will take to complete</td>
</tr>
</tbody>
</table>
Estimating Example

- Epic: As a Dinner host I would like to have a clean kitchen so guests feel comfortable eating the prepared food
  - User Stories
    - As a dinner host I would like to have clean floors so guests are comfortable in the kitchen
    - As a dinner host I would like clean appliances so that the food will be prepared on sanitary appliances
    - As a dinner host I would like clean dishes and utensils so that I have everything necessary to prepare, cook and serve the food in.
    - As a dinner host I would like to have clean countertops and cabinets so that the prep surfaces are sanitary.
• Clean Floors
  - Sweep Floor
  - Mop Floor

• Clean Appliances
  - Clean Oven
  - Clean out Refrigerator
  - Clean Stove top
  - Wipe Refrigerator and Dishwasher

• Clean Dishes
  - Scrub Pots and Pans
  - Wash dishes
  - Put dishes away

• Clean Countertops and Cabinets
  - Wipe countertops
  - Wipe cabinets and drawers
  - Dust light fixtures
  - Clean out sink