ISA 201
Information System Acquisition
Lesson 10
Software Development
Learning Objectives

Today you will learn to:

Overall: Given a DoD IT/SW acquisition scenario, recommend the appropriate software development methodology/mix of methodologies to ensure program success.

- Identify the most appropriate software methodology (or a combination of methodologies) to meet the expectations of the government.
- Identify the key principles of an agile software development process.
- Describe selected aspects of the SCRUM method.
- Develop estimates for tasks using agile techniques and Story Points.
- Recognize the purpose of key agile software measures.
- Define the role of 'Software DEVelopment and information technology OPerationS ' (DevOps) in a software development project.
• Software Development Life Cycle Models
  • Agile Software Development Overview
  • Scrum
  • Agile Measures
  • DevOps
• 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
Software is Here to Stay!

Aircraft

- F-22 Raptor (2005)
- Boeing 787 (2009)
- F-35 Fighter (2015?)

Operating Systems

- Windows XP (2001)

Automobiles

- Chevy Volt (2010)
- Software in Typical New Car (2013)

Web Pages

- Facebook (without backend code)
Software Development Life Cycle Models

• Traditional
  - Waterfall
  - Incremental
  - Spiral

• Agile
  - Scrum
  - Extreme Programming (XP)
  - Adaptive Software Development
  - Dynamic System Development Method
  - Rapid Application Development
Answer these questions based on the Homework:

• Which process model is best suited for when all the requirements are understood up front and are stable?
• Which process model might be best to use on a project with a lot of risks?
• Which process model would you choose if the requirements are not well understood and are emerging?
• Which process should I consider if the requirements are known up front and immediate functionality is needed?
Lesson Overview

- Software Development Life Cycle Models

Agile Software Development Overview

- Scrum
- Agile Measures
- DevOps
Agile Terms

- Agile Manifesto
- Backlog
- Backlog Grooming
- Burn down
- Burn up
- Cumulative Flow Diagram
- Daily Standup
- Definition of “Done”
- DevOps
- Empirical
- Feature Toggles
- Incremental
- Iterative
- Kanban
- Minimal Viable Product
- Pair Programming
- Planning Poker
- Product Owner
- Refactoring
- Retrospective
- Scrum
- Scrum Master
- Spike
- Sprint
- Story Points
- TDD
- User Story
- Velocity
- WIP
- YAGNI
## CHAOS Resolution by Agile versus Waterfall

<table>
<thead>
<tr>
<th>SIZE</th>
<th>METHOD</th>
<th>SUCCESSFUL</th>
<th>CHALLENGED</th>
<th>FAILED</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Size Projects</td>
<td>Agile</td>
<td>39%</td>
<td>52%</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>Waterfall</td>
<td>11%</td>
<td>60%</td>
<td>29%</td>
</tr>
<tr>
<td>Large Size Projects</td>
<td>Agile</td>
<td>18%</td>
<td>59%</td>
<td>23%</td>
</tr>
<tr>
<td></td>
<td>Waterfall</td>
<td>3%</td>
<td>55%</td>
<td>42%</td>
</tr>
<tr>
<td>Medium Size Projects</td>
<td>Agile</td>
<td>27%</td>
<td>62%</td>
<td>11%</td>
</tr>
<tr>
<td></td>
<td>Waterfall</td>
<td>7%</td>
<td>68%</td>
<td>25%</td>
</tr>
<tr>
<td>Small Size Projects</td>
<td>Agile</td>
<td>58%</td>
<td>38%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Waterfall</td>
<td>44%</td>
<td>45%</td>
<td>11%</td>
</tr>
</tbody>
</table>

The resolution of all software projects from FY2011-2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

• **Deliver performance at the speed of relevance.**
  - Success no longer goes to the country that develops a new technology first, but rather to the one that better integrates it and adapts its way of fighting. **Current processes are not responsive to need;**
  
  Reference: Secretary of Defense Mattis NDS 2018

• Dr. Will Roper, Assistant Secretary of the Air Force for Acquisition, Technology and Logistics, July 2018
  - “In a future war we could be changing software every day as a necessary factor for winning,”
Software in the DoD Acquisition Environment

- 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
Expectations for System level SE Technical Reviews Given Incremental Software Development

- Develop the minimum viable requirements
- Define configuration item level requirements for the build(s) or increment under review
- Develop a minimum viable architecture that consists of an initial software architecture and design
- Document expectations for lower-level component artifacts and a minimum set of characteristics that defines the level of tailoring and acceptance criteria for these artifacts.
- Conduct a risk assessment that covers the full scope of the system;
- Define progress and product metrics for iterations/builds and total system development.
DAG: Example Implementation of PDR for Incremental SW Development

- Multiple Build-level Reviews
- Evolving lower-level requirements and design maturity
- Requirements at an initial PDR may be fully defined for a set of capabilities, with future builds fully defining low-level requirements for additional capabilities
- Allocated Baseline may not be fully established at initial PDR
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 and end-users. 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  
Scaled Agile Framework (SAFe)  
Large-Scale Scrum (LeSS)  
Disciplined Agile Delivery (DAD)
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*.
“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.
• Software Development Life Cycle Approaches
• Agile Software Development Overview

• Scrum
• Agile Measures
• DevOps
Exercise: Scrum Ball

This exercise will help us put in practice some of the topics we have been discussing with Agile. We will implement self-organizing teams, practice continuous improvement, time boxed iterations, and utilize empiricism.

**Rules**

- **Objective:** Get as many balls from start to finish through the team as possible in 2 minutes
  - 1 Ball = 1 Point
  - Each ball must have air-time during each pass
  - Each ball must be touched at least once by every team member
  - The ball cannot be passed to the team member on your immediate left or right
  - Each ball must return to the same person who placed it into the system. Start Point = End Point
  - Balls may be reused
  - Dropped ball equals 0 Points and stays on the ground
  - Teams consist of 5+ members

**Game Play**

1. The teams will get two minutes of preparation time to organize themselves and estimate how many balls they can pass through the system
2. Each team will write their estimate on the designated whiteboard
3. Play a two minute iteration of the game
4. Each team will write their actual score on the designated whiteboard
5. The teams will get one minute to perform a retrospective to discuss how to improve the process and develop a new estimate
6. Repeat steps 2-5 for a total of 5 iterations.
Scrum Ball: Debrief

• How accurate was the first estimate? Did the estimates improve?
• What was the team leadership model?
• How did you achieve dramatic increase in velocity?
• Did every experiment work?
Scrum is an iterative and incremental software development process framework. It relies on collaborative, self-organizing, cross-functional, self-managing teams. The empirical approach to managing the process is responsive to emerging or changing requirements. It 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
  - Sprint Planning
- Daily
  - Daily Scrum
  - Implementation
- End
  - Sprint Review
  - Sprint Retrospective

“Done” Software
- Velocity

Scrum Key
- Events
- Artifacts
- Measures
- Other

Sprint Backlog
Burndown
Sprint Planning

- **Who**
  - Development 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
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
- TimeBoxed
  - 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 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
- Refines the backlog periodically to ensure priorities are correct, items are at the 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 their 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 Artifacts

<table>
<thead>
<tr>
<th><strong>Product Backlog</strong></th>
<th><strong>Sprint Backlog</strong></th>
<th><strong>Done Software</strong></th>
</tr>
</thead>
</table>
| - Prioritized list of user stories, features, or work items detailing the desired functionality for the product being developed. | - 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” 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 |
| - 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 | | |
**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:** 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.
User Story - Common Format

User Story is a description consisting of one or more sentences in the everyday or business language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function. [Wikipedia]

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

• Conversations
  - Flesh out the details
  - Ensure that the intent of the requirement is understood

• Acceptance Criteria
  - Defines what constitutes when the story is complete
  - Used to create test cases

As a mission planner, I want to plan my route so that I can see how much fuel will be needed for the flight.
Agile Estimation: Story Points

- **Story Point** - Unit of measure for evaluating the relative size of an epic, feature, or user story.
- **Typical Estimation Scale** roughly follows the Fibonacci sequence
  - 0, 1, 2, 3, 5, 8, 13, **20, 40, 100**
  - Some teams include ½ and ∞
- **Delphi / Expert Judgement** method for estimating
  - 2 Common Agile Estimation Techniques
    - Planning Poker
    - Affinity Estimation
- A user story assigned a two should be roughly twice the size of a user story assigned a value of one
- Unit-less Measurement
Story Points

Story Point Scale

1 2 3 5 8 13 20
Each team member has a set of valid Story Point Cards

Each Product Backlog Item (PBI) is discussed briefly,

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

### Planning Poker Card(s) Interpretation

<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>
As the facilitator for a planning poker session, you’ve given each of your six team members cards labeled, 1, 2, 3, 5, 8, and 20. Each team member chooses a card indicating the number of story points to assign to a particular user story. The resulting values are 3, 5, 8, 8, 13, and 20. What should you encourage the team to do next?

a. Use a different set of numbers
b. Ask the team members who chose the values of 3 and 20 to explain their choices
c. Calculate the average number of points and assign it to the user story
d. Decide what story point value to assign to the user story on behalf of the group
Affinity Estimation Variation

- Use columns going from smallest to largest as you move left to right.
- Identify the smallest candidate PBI and place in the far left column.
- Identify the largest PBI and place in the far right column.
- Compare each PBI to the PBI in a column if smaller go left if larger go right.
- Once all PBIs are in columns estimate the story points for each column.
- Record estimate for each PBI.

Whiteboard, Wall, or Bulletin board.
Estimating Example

- **Epic:** As a Dinner host I would like to have a clean kitchen so guests feel comfortable eating the prepared food
  - Decomposed into 4 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 with 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.
    - 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
Affinity Estimating Example

- Smallest
  - Clean Floors
    - Clean Counters
    - 3 pts
  - Clean Dishes
    - 5 pts

- Largest
  - Clean Appliances
    - 13 pts
• Software Development Life Cycle Approaches
• Agile Software Development Overview

• Scrum
  - Estimating Exercise (go to Exercise Folder)
• Agile Measures
• DevOps
The objective of this exercise is let each team practice estimating using story points. Each team will create estimates for each of the books based on how long it would take to read and comprehend the book using story points.

<table>
<thead>
<tr>
<th>Title</th>
<th>Story Point Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Great Gatsby</td>
<td></td>
</tr>
<tr>
<td>SAFe Distilled</td>
<td></td>
</tr>
<tr>
<td>The Eye of the World</td>
<td></td>
</tr>
<tr>
<td>Lean Startup</td>
<td></td>
</tr>
<tr>
<td>Refactoring</td>
<td></td>
</tr>
<tr>
<td>Agile Software Development with Scrum</td>
<td></td>
</tr>
<tr>
<td>Introduction to Algorithms</td>
<td></td>
</tr>
<tr>
<td>Extreme Programming Explained: Embrace Change</td>
<td></td>
</tr>
<tr>
<td>The DevOps Handbook</td>
<td></td>
</tr>
<tr>
<td>Introduction to Disciplined Agile Delivery</td>
<td></td>
</tr>
<tr>
<td>Release It!</td>
<td></td>
</tr>
<tr>
<td>Kanban</td>
<td></td>
</tr>
<tr>
<td>Agile Estimating and Planning</td>
<td></td>
</tr>
</tbody>
</table>

- Each team member has a set of valid cards
- Read the book info and discuss briefly
- Each team member selects the card that matches their estimate
- Everybody shows cards at the same time
- Discuss the outliers and why they had their estimate
- Re-estimate until you have a convergence
- Record your estimate for each book
Affinity Estimation Variation
Teams 1, 4

- Read the book info and discuss briefly
- Use columns going from smallest to largest as you move left to right
- Identify the smallest candidate PBI and place in far left column
- Identify the largest PBI and place in far right column
- Compare each PBI to the PBI in a column if smaller go left if larger go right
- Once all PBIs are in columns estimate the story points for each column
- Record estimate for each PBI

Whiteboard, Wall, or Bulletin board
Scrum Summary – Key Components of the Scrum Framework

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

- **3 Roles**
  - Product Owner
  - Development Team
  - Scrum Master

- **3 Artifacts**
  - Product Backlog
  - Sprint Backlog
  - “Done” Software
• Software Development Life Cycle Approaches
• Agile Software Development Overview
• Scrum

• Agile Measures
• DevOps
• Measurement is also used in agile approaches - generally the measures are used internally by the development teams

• The next few slides:
  - Introduce measures used in Agile development
  - Teach how to interpret Agile measurements
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 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
Typical Measures for Agile Development

- Kanban Board for Task Tracking
- Velocity
- Sprint Burn-down Charts
- Release Burn-Up Charts
- Cumulative Flow Diagrams

Teams focus on delivering working code
- Used to track tasks and their current status visually
- Taken from Lean they represent the workflow
- Vary from very simple to complex
- Cumulative Workflow diagrams would represent every state from the Kanban Board
- Numbers represent Maximums for that workflow (i.e. WIP is limited to 5 items at any one time)
- Limits encourage swarming to prevent bottlenecks and to get features all the way through the system
• 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!
• 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
• Sprint Burndown can be represented as either remaining hours or remaining story points
Release Burnup Chart

Fixed Scope

Estimated Completion Date

Planned

Actual

Software Development 54
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.

64 Total “items” in the Product Backlog

64-36 = 28 items remain in ToDo state

23 Total “items” in the Done state

Y = Current “items” in Progress (WIP)
(36-23 = 13)

X = WIP Queue Duration (Lead Time)

Y = WIP Queue Length (Number Items in Progress)
Lesson Overview

Lesson Plan

• Software Development Life Cycle Approaches
• Agile Software Development Overview
• Scrum
• Agile Measures
• DevOps
Silos and Delivery Challenges

1st Gap: Addressed by Agile Dev
2nd Gap: Addressed by DevOps
DevOps is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) operations professionals.

DevOps integration targets product delivery, quality testing, feature development and maintenance releases in order to improve reliability and security and faster development and deployment cycles.

- **DevOps aims to help an organization rapidly produce software products and services.**
- **DevOps integrates developers and operations teams to improve collaboration and productivity.**
How to fix these problems? Origin of “DevOps”

- Develop and test against production-like environments
- Iterative and frequent deployments using repeatable and reliable processes
- Continuously monitor and validate operational quality characteristics

Track and Plan everything
Version everything
Automate everything
Test everything
Audit and Monitor everything
Dashboard everything
Roles in a DevOps Environment

Authorizing Official
Program Management/Systems Engineering

Planning

Feedback

Use, Operate and Monitor

Inception

User Stories

Features and Design

Construction

Features and Design

Release

Release Manager

Typical Agile Roles (Product Owner, Facilitator, Users, Team Members, Stakeholders)

Construction

Integration & Test

Acceptance

Acceptance

Government Test Organizations

Test Automation Manager

New roles
DevSecOps strives to automate core security tasks by embedding security controls and processes into the DevOps workflow. DevOps originally focused primarily on automating code security and testing, but now it also encompasses more operations-centric controls.
## DevOps High Performers

<table>
<thead>
<tr>
<th>Company</th>
<th>Deploy Frequency</th>
<th>Deploy Lead Time</th>
<th>Reliability</th>
<th>Customer Responsiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>23,000 / day</td>
<td>minutes</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Google</td>
<td>5,500 / day</td>
<td>minutes</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Netflix</td>
<td>500 / day</td>
<td>minutes</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Facebook</td>
<td>1 / day</td>
<td>hours</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Twitter</td>
<td>3 / week</td>
<td>hours</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Typical Enterprise</td>
<td>1 / 9 months</td>
<td>months</td>
<td>Low to Medium</td>
<td>Low to Medium</td>
</tr>
</tbody>
</table>

Reference: The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win
• 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
Additional Resources

- DAU – CLE 076
- DAU Agile Workshop
- ISA230 Agile Acquisition
- 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
- OMB: Contracting Guidance to Support Modular Development
- GAO Agile Guide (Coming Soon)
Overall: Given a DoD IT/SW acquisition scenario, recommend the appropriate software development methodology/mix of methodologies to ensure program success.

- Identify the most appropriate software methodology (or a combination of methodologies) to meet the expectations of the government.
- Identify the key principles of an agile software development process.
- Describe selected aspects of the SCRUM method.
- Develop estimates for tasks using agile techniques and Story Points.
- Recognize the purpose of key agile software measures.
- Define the role of 'Software DEVELOpment and information technology OPERationS' (DevOps) in a software development project.