Agenda

- Iteration 1 – Introduction and Agile Overview
- Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
- Iteration 3 – Release Planning
- Iteration 4 – Writing User Stories
- Iteration 5 – Estimating User Stories
- Iteration 6 – Sprint Planning
- Iteration 7 – Sprint Execution
- Iteration 8 – Sprint Demo, Retrospective, and Delivery
- Iteration 9 – Agile Trends and Resources
Agenda

• Iteration 1 – Introduction and Agile Overview
  – Getting to Know You, Training Outcomes and Expectations
  – Defining an Agile Environment and Review of Agile Methods
• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
• Iteration 4 – Writing User Stories
• Iteration 5 – Estimating User Stories
• Iteration 6 – Sprint Planning
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery
• Iteration 9 – Agile Trends and Resources
Getting To Know You!

Mr. Steve Baynes

• Department manager for Northrop Grumman’s process-oriented commercial software product, e.POWER®
• Certified ScrumMaster, member of the Agile Alliance organization and speaks often on Agile development
• About 8 years ago, Steve and his team got on board with the Agile Manifesto and Agile Principles
• Leveraging Agile principles and practices, Steve works with business development, project implementation teams and customers to continually improve the e.POWER product from both a feature and quality perspective
Today’s Outcomes

Today you will…
• Develop an understanding of the major agile engineering practices
• Participate in a planning and estimating scenario
• Gain insight into Agile testing principles
• Participate in a team retrospective
• Have fun working together!
Today’s Plan

Collaborate

Share

Brainstorm

Have Fun!

Ask
Expectations

What are your expectations?

What questions do you have?

Time: 5 minutes
What is your level of experience?

Scale of 1 to 5
Agenda

• Iteration 1 – Introduction and Agile Overview
  – Getting to Know You, Training Outcomes and Expectations
  – Defining an Agile Environment and Review of Agile Methods

• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
• Iteration 4 – Writing User Stories
• Iteration 5 – Estimating User Stories
• Iteration 6 – Sprint Planning
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery
• Iteration 9 – Agile Trends and Resources
Agile Engineering

• Adaptive, Responsive, Continuously Learning, Evolving
• Inspects and adapts to change
• Includes the entire product life cycle
• Focuses on the value stream
Agile Engineering

• Delivers value to the customer through frequent and short iterations (sprint)

• Incremental release of capabilities; each iteration (sprint) is potentially releasable

• Do a little bit of everything every iteration (sprint)
  – Plan, Test, Design, Build
Recognizing the Need for Change

GAO Highlights

Software Development
Effective Practices and Federal Challenges in Applying Agile Methods

What GAO Found

GAO identified 32 practices and approaches as effective for applying Agile software development methods to IT projects. The practices generally align with five key software development project management activities: strategic planning, organizational commitment and collaboration, preparation, execution, and evaluation. Officials who have used Agile methods on federal projects generally agreed that these practices are effective. Specifically, each practice was used and found effective by officials from at least one agency, and ten practices were used and found effective by officials from all five agencies. The ten practices are:

1. Start with Agile guidance and an Agile adoption strategy.
2. Enhance migration to Agile concepts using Agile terms, such as user stories (used to convey requirements), and Agile examples, such as demonstrating how to write a user story.
3. Continuously improve Agile integration and organization level.
4. Seek to identify and address Agile barriers.

GAO identified 14 challenges with adapting and applying Agile in the federal environment (see table).

Table: Federal Challenges

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teams had difficulty collaborating closely</td>
<td>Procurement practices may not support Agile projects.</td>
</tr>
<tr>
<td>Teams had difficulty transitioning to self-directed work</td>
<td>Customers did not trust iterative solutions.</td>
</tr>
<tr>
<td>Staff had difficulty committing to more timely and frequent input</td>
<td>Teams had difficulty managing iterative requirements.</td>
</tr>
<tr>
<td>Agencies had trouble committing staff.</td>
<td>Compliance reviews were difficult to execute within an iteration time frame.</td>
</tr>
<tr>
<td>Timely adoption of new tools was difficult.</td>
<td>Federal reporting practices do not align with Agile.</td>
</tr>
<tr>
<td>Technical environments were difficult to establish and maintain</td>
<td>Traditional artifact reviews do not align with Agile.</td>
</tr>
<tr>
<td>Agile guidance was not clear.</td>
<td>Traditional status tracking does not align with Agile.</td>
</tr>
</tbody>
</table>

Source: GAO

http://gao.gov/products/GAO-12-681
Agile Manifesto and Principles

**Manifesto**

- **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.

**Principles**

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.
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.
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.
Agile Methods

Agile is about how we believe people are best motivated to do work and about demonstrating high value on a regular basis particularly in environments that face high requirement volatility and unpredictability.
Agile Trends and Adoption

Agile Keywords & Phrases

- Each Project consists of one or more **Releases**
- Each Delivery (Release) consists of one or more **Sprints**
  - Each Sprint consists of one or more **Stories**
  - Each Story consists of **Tasks**

2-9 months
Multiple releases per spin
Product Burndown

1-4 weeks
Fixed time
Potentially deliverable
Checkpoint
Sprint burndown

User Capability
Measures
work size
(points)

Technical reviews
implementation,
verification,
documentation, etc.

**Deliver working functionality every Sprint**
The Scrum Framework

Capabilities and User Stories
Prioritized by Product Owners
Creates the release plan

The Daily Tasks managed by the Cross Functional Team

Design, Code, Integrate, Test

Commitment
Identification of Impediments
Communication

Release Plan

Product Backlog

Sprint Backlog

24 Hours
Daily Scrum Meeting

2-4 Week Sprint

Feature Demonstration and Retrospective

Potentially Shippable Product Increment

System Architecture
System Design Requirements

Inspect and Adapt
Visibility and Transparency

Image from: Mike Cohn, Mountain Goat Software
Reference: Ken Schwaber, Jeff Sutherland
The objective of the Ball Point game is to get as many balls through the team as possible within two minutes.

Each ball must be touched at least once by every team member and must end with the same person with whom it began.

After two minutes the team is allowed an additional minute to discuss the process and how it could be improved. The game is played a total of five times.
Tek Talk: Ball Point Game

Rules of the Game

• You are one big team
• The object of the game is to process the most work possible by passing balls through the pipeline.
• Balls must pass between people in the air
• You cannot pass a ball to your neighbour
• Work is completed when a ball arrives back at the start point after being touched by everybody on the team
• There will be 5 iterations of 2 minutes each
• Retrospectives and next iteration planning are 1 minute
• Each iteration needs an estimate for the number of balls to be completed
**What Did We Demonstrate?**

The inspect and adapt cycle can be used to improve a system of production.

A system has a natural velocity.

Velocity of a team stabilizes because of the team’s system and because the team learns how to work together.

If you want to go faster, you have to change the system.

100% utilization is not the same as maximum throughput through the system.

Having a large team makes it harder to hear everybody’s input.
Agenda

• Iteration 1 – Introduction and Agile Overview

• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
  – Creating the Agile Project Team
  – Defining the Business Strategy (knowing where you're going before release planning)

• Iteration 3 – Release Planning

• Iteration 4 – Writing User Stories

• Iteration 5 – Estimating User Stories

• Iteration 6 – Sprint Planning

• Iteration 7 – Sprint Execution

• Iteration 8 – Sprint Demo, Retrospective, and Delivery

• Iteration 9 – Agile Trends and Resources
## Roles and Responsibilities

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
</table>
| **Chief Engineer/Architect** | - Ensures the integrity of the architecture  
  - Communicates the systems design  
  - Has no predefined team  
  - Adds to the Product Backlog |
| **Product Owner**           | - Defines the features of the product  
  - Manages project features and release to optimize return on investment (ROI)  
  - Prioritizes features according to mission/user needs  
  - Inspects increment and makes adaptations to project  
  - Can change features and priority every sprint  
  - Communicates project progress and status  
  - Accepts or rejects work |
| **Team**                    | - Cross-functional, seven plus/minus two members  
  - Selects the sprint goal and specifies work results  
  - Commits to what it feels it can accomplish  
  - Has authority to do everything within existing standards and guidelines to reach the sprint goal  
  - Manages itself and its work  
  - Collaborates with Product Owner to optimize value  
  - Demos work results to the Product Owner |
| **ScrumMaster**             | - Ensures that the team is fully functional, productive and improves quality  
  - Enables close cooperation across all roles and functions and removes barriers  
  - Shields the team from external interferences  
  - Ensures that the process is followed  
  - Teaches Product Owner and Team how to fulfill their roles  
  - Does not make decisions for the team |
Transition from functional grouping to cross functional teams

Isolated progress with too many hand-offs and barriers
Project Team Structure (An Example)

- PM and Technical Lead
- Chief Engineer
- Chief Architect
- Quality

- Cross Functional Team 1
- Cross Functional Team 2
- Cross Functional Team 3
- Cross Functional Team n

Services

Supports Cross Functional Teams
- Push accountability and ownership to the team level
- Everyone trained

Network/ Systems Administration
Configuration Mgt.
Cross Functional Teams
Agenda

- Iteration 1 – Introduction and Agile Overview
- Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
  - Creating the Agile Project Team
  - Defining the Business Strategy (knowing where you're going before release planning)
- Iteration 3 – Release Planning
- Iteration 4 – Writing User Stories
- Iteration 5 – Estimating User Stories
- Iteration 6 – Sprint Planning
- Iteration 7 – Sprint Execution
- Iteration 8 – Sprint Demo, Retrospective, and Delivery
- Iteration 9 – Agile Trends and Resources
High Level Agile Stages
5 Levels of Planning


Capabilities identified
Some initial analysis
Product Roadmap Workshop

Product Roadmap provides the vision usually 9-12 months

INPUT
- Business/Mission Objectives
- Technology Vision

OUTPUT
- High-Level product capabilities mapped to several releases
- Priorities
- Resources
- Product Backlog updated

Defining the Product Roadmap

Product Owner, Customers, Management
Product Roadmap

Product: Hotel Website for RestEZ

<table>
<thead>
<tr>
<th>Release 1</th>
<th>Release 2</th>
<th>Release 3</th>
<th>Release 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room reservations and payment</td>
<td>Conference offerings</td>
<td>Special discounts for room reservations</td>
<td>Air and hotel package deal</td>
</tr>
<tr>
<td>User profiles for future visits</td>
<td>Online chat support</td>
<td>Improve usability</td>
<td>Meeting and Business plans</td>
</tr>
<tr>
<td>Hotel amenities</td>
<td>Local information</td>
<td>Google maps</td>
<td>and reservations</td>
</tr>
</tbody>
</table>
Release 0: Project Start up

• Start building the team.
  – Begin with at least one or two senior developers, the Scrum Master and Product Owner and one or more stakeholder representatives.
  – Training

• Create an initial architecture for the system.
  – You need to have at least a general idea of how you're going to build the system.
  – Identify an architectural strategy. Work through the design details later during future Sprints in model sessions.
  – Every Sprint must deliver at least some piece of business functionality

• Setting up the environment.
  – You need workstations, development tools, and work areas. Start with just enough to get the team going and continue to build on this in future releases.

• Determine first release date and Sprint length

• Create a transition backlog
The length of a Release may vary while the length of a Sprint remains constant.
Tek Talk: Team

• Reflecting on the section *Creating the Team* address the following.

• **Discussion**: Whose responsibility?
  
  – The Product Owner is micromanaging the team making self-managing impossible.
  
  – The team is struggling to understand the priorities of the work.
  
  – A team member is constantly late for the daily standup.
  
  – The team is not able to deliver on their commitments.

Time: 5 minutes
Agenda

• Iteration 1 – Introduction and Agile Overview
• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
  – What we are going to get done over the next several months
• Iteration 4 – Writing User Stories
• Iteration 5 – Estimating User Stories
• Iteration 6 – Sprint Planning
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery
• Iteration 9 – Agile Trends and Resources
Define the Release and Sprint Cycle

Example

3 Month Release Cycle

Planning started in Dec.

1/07/13

Sprint 1

2/01/13

Sprint 2

3/01/13

Sprint 3

3/29/13

Release

Day 1
A.M. = Release Planning Mtg
P.M. = Sprint1 Planning Mtg

End of Each Sprint
Retrospective
Demonstration and Review
Potentially releasable
Update Metrics

Day 1 of Each Sprint
Sprint Planning Meeting
2-4 hours

Formal release/Delivery
The Product Backlog

- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner

Product Vision
Customer Needs

As a vacationer, I want to search room availability...
As a vacationer, I want to change my reservation...
As a vacationer, I want to cancel my reservation...
As a vacationer, I want to pay with a credit card...

Ranked according to Business value
The Product Backlog

Product: Hotel Website for RestEZ

Use Case Flow: Make Room Reservations

<table>
<thead>
<tr>
<th>User Stories</th>
<th>Acceptance Criteria</th>
<th>Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a vacationer, I want to search room availability...</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>As a vacationer, I want to save my request...</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>As a vacationer, I want to pay with a credit card...</td>
<td>• Demo with Visa</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>• Demo with AmEx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Demo 3....</td>
<td></td>
</tr>
</tbody>
</table>

The Release Plan

What is our velocity? How many story points can we take on for this release? What can we commit to?

Ranked according to Business Value

Owned by the Product Owner with the opportunity to reprioritize each Sprint
Velocity (Based on history)

- Velocity is the amount of work a development team completes in a Sprint (story points completed)
- Velocity is a range; Look for the high, the low, and the mean.

**Team A Velocity**
- High: 45 story points
- Low: 30 story points
- Mean: 37 story points

**Project Velocity per Sprint**
- High: 155 story points
- Low: 120 story points
- Mean: 137 story points
Release Planning Meeting

- Meeting is time-boxed
- Usually ½ - 1 day depending
- Occurs with the entire project team
Commit to the Release Plan

- Capabilities/Goals identified
- High level requirements and initial user stories mapped
- User stories (functional and non-functional requirements)
  - Ex.: Project Teams average about 137 user story points per Sprint; for a release with 6 Sprints this is about 900 story points. The scope is 720-930 user story points of work.
  - Total number of user stories planned (125)
- Dependencies identified
- Known or assumed velocity by development team and project team
- Total number of user stories planned
- Planned hours (WBS element)
Agenda

• Iteration 1 – Introduction and Agile Overview
• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
• Iteration 4 – Writing User Stories
  – Three critical aspects: Card, Conversation, and Confirmation
• Iteration 5 – Estimating User Stories
• Iteration 6 – Sprint Planning
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery
• Iteration 9 – Agile Trends and Resources
A User Story is Comprised of…

• *A written description* of the story, used for planning and as a reminder

• *Conversations* about the story that serve to flesh out the details of the story

• *Tests* that convey and document details that can be used to determine when a story is complete
Tek Talk: Requirements. It’s about the conversation

- Exercise on understanding what the customer wants.
- Follow instructions provided by the Agile Trainer.
- You will need to work in pairs to complete this activity.

Time: 10 minutes
• Requirements might say
  – The product shall have a gas engine
  – The product shall have four wheels
    • The product shall have a rubber tire mounted to each wheel
  – The product shall have a steering wheel
  – The product shall have a steel body

Source Mike Cohn: www.mountaingoatsoftware.com
As a <lawn service provider> I want to mow lawns quickly and easily.

As a <lawn service provider> I want to sit comfortably while mowing lawns.

Reference: Mike Cohn, mountaingoatsoftware.com
User Stories

- **Functional stories**
  - often based off a scenario of a use case
  - On large projects a user can be another system

- **Non-functional stories**
  - Includes things such as reliability, availability, maintainability, sustainability..

- **Spike Stories**

**Note:**

- Needs a Definition of Done
  - Design, Write tests, code, unit tests, documentation, etc.

- No credit for partial work – either done or not done

- Includes Acceptance Criteria

**User Story**

As vacationer, I want to search for available rooms.

**Acceptance Criteria**

- Search by location
- Search by date
- Search on type

Include critical information as needed:
- Design Notes, Assumptions, Constraints
Writing User Stories

- Often written by the Product Owner or as a team
- Brainstorm to generate ideas
- Some stories start out as epic stories; break them down
- Stories should be drafted and estimated prior to the release planning meeting

INVEST in good User Stories.....

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable
The hotel reservation system shall enable the user to complete a reservation transaction.

As a premium member, I want to search for available discounted rooms so I can save money while reserving the room of my choice.

As vacationer, I want to search for available rooms so I can see what is available and compare rooms.

As vacationer, I want to save my selections so I can come back later and make changes or complete my reservation.

STEP #1 Write User Stories
## Requirement to User Stories with Acceptance Criteria

The hotel reservation system shall enable the user to complete a reservation transaction.

<table>
<thead>
<tr>
<th>User Story</th>
<th>Acceptance Criteria and Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a vacation planner I want to pay for my reservation with a credit card so I can confirm my reservation.</td>
<td></td>
</tr>
</tbody>
</table>
* Demonstrate with American Express  
* Demonstrate with Master Card  
* Demonstrate with Visa |

**STEP #2**

Write Acceptance Criteria
### Stories for non-functional requirements

<table>
<thead>
<tr>
<th>User Story</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>As a vacationer and user of the hotel website, I want the system to be available 99.99% of the time so I can search the website for information any time...</strong></td>
<td></td>
</tr>
<tr>
<td><strong>As vacationer, I want web pages to download in &lt;4 seconds so I don’t have to wait!!</strong></td>
<td></td>
</tr>
<tr>
<td><strong>As the hotel website owner, I want 1,000 concurrent users to be able to access the site at the same time with no impact to performance so users are not turned away due to speed.</strong></td>
<td></td>
</tr>
</tbody>
</table>

Reference: Mike Cohn, mountaingoatsoftware.com

**Describes system behavior or characteristics**

**STEP #3 Identify constraints**
Creating User Stories

Information from each Sprint feeds into the next

Slices cut across all sub-systems

High-level Analysis and Design (on-going)

Presentation / User Interface

Business Logic / Services

Database / Integration

Sprint 1  Sprint 2  Sprint 3  Sprint 4  Sprint 5  Sprint 6  Sprint 7  Sprint 8

Release 1  Release 2
A “story” or system capability often spans several functional areas.
• Scenario: Based on customer needs, your team has defined a logical architecture for an online hotel reservation system.

The system is a traditional 3 tier architecture:
- a database layer (to persist reservations),
- a business logic layer (to manage reservations),
- and a browser-based user interface (to receive customer input).

Your product owner started to create the product backlog and has provided the following Epic:

As a “vacationer” I want to search the hotel’s facilities and related amenities.

• As a group we will write 1 story for the epic story above including acceptance (testing) criteria.

Time: 5 minutes
 Agenda

• Iteration 1 – Introduction and Agile Overview
• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
• Iteration 4 – Writing User Stories
• **Iteration 5 – Estimating User Stories**
  – Estimating the effort to develop each user story
• Iteration 6 – Sprint Planning
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery
• Iteration 9 – Agile Trends and Resources
Estimating User Stories

- Traditional estimates focus on absolutes
- Agile Estimates focus on relative size estimates

Why are relative size estimates preferred?
- People are better at comparing versus measuring in absolutes
- Estimates are faster
- Basic math still applies
- Easier to reach consensus

Reference: Rally Software
Recommendations for Estimating User Stories

• Keep your estimates high-level intuitive guesses
  – Don't over-analyze the details
• Estimating using a number sequence (Fibonacci numbers)
  – It get less precise as the estimates get bigger, which builds a natural distribution curve into your estimates.
• Estimate as a team
• Track your Velocity over time
• You can also track your reliability
  – i.e. the number of points delivered as a percentage of the number of points committed to at the start of the Sprint.
• At the start of each Sprint, look back at your Velocity
• Don't try to reconcile points with hours
• Commit as a team
What could these story points possibly mean?

1: Trivial. This will be an easy point for my burn-down chart. (Low uncertainty, low complexity, and low effort)

2: A little bit of thought is required, but I’ve done so many, I won’t even break a sweat. Or, it sounds trivial and I’m going to hedge my bets a bit.

3: I’ve done this a lot, I know what needs to be done, maybe a few extra steps, but that’s it. Doubtful that I’ll need to Google anything.

5: OK, now we’re getting into the heart of the problem or, I don’t do this very often. I’ll need to think through this a bit. (At least one of the variables (complexity, effort, uncertainty has increased)

8: This is going to take some time and planning. Or, I’ve seen other people do this and I prototyped something like it a few years ago.

13: Possibly the maximum amount I can do and still get it done in one sprint. If this item is truly reduced to its simplest form, then it’s a complex piece of work. We need to consider our velocity.

21: This is very complex or contains a high level of uncertainty and/or effort.

?: Large. Needs more information and possibly broken down into several stories.
Estimating the user stories for a release. A release is one or more Sprints.

Going into the estimation phase, stories for the release have been identified and each has verification objectives; Stories have been discussed with the team.

Steps
- Each estimator is given a deck of cards, each card has a valid number such as (1, 2, 3, 5, 8, 13, 21, ?)
- The teams read the stories
- An “average” story is selected (not the hardest, not the easiest)
- The story is read to the team and discussed briefly
- Each estimator selects a card to reveal his estimate
- Cards are turned over so everyone can see them
- Differences in estimates are discussed; especially outliers
- Re-estimate until estimates converge

Reference: www.mountaingoatsoftware.com
<table>
<thead>
<tr>
<th>Backlog Item</th>
<th>Relative Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Putting away a load of laundry</td>
<td></td>
</tr>
<tr>
<td>Building a shed</td>
<td></td>
</tr>
<tr>
<td>Vacuuming the floors</td>
<td></td>
</tr>
<tr>
<td>Packing lunch</td>
<td></td>
</tr>
</tbody>
</table>

Time: 10 minutes
Agenda

• Iteration 1 – Introduction and Agile Overview
• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
• Iteration 4 – Writing User Stories
• Iteration 5 – Estimating User Stories
• Iteration 6 – Sprint Planning
  • What we are going to get done over the next few weeks
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery
• Iteration 9 – Agile Trends and Resources
High Level Agile Stages

- Vision Customer Needs
- Product Roadmap
- Release Planning
- Sprint Planning
- Sprint Execution
- Sprint Demo and Retrospective
- Delivery
- Product Backlog
Meeting is time-boxed
Usually ½ day depending on length of the Sprint

INPUT
- User stories with business value
- User stories with estimates
- Team capacity
- Team velocity

Sprint Planning Meeting

OUTPUT
- Sprint Goals
- Stories for the Sprint
- Tasks with hours for each user story
## Release Plan to Sprint Plan

### Example: Hotel Website

#### Release Plan

<table>
<thead>
<tr>
<th>Business Value</th>
<th>User Story</th>
<th>Test Objectives</th>
<th>Estimate</th>
</tr>
</thead>
</table>
| 580            | As a vacationer, I want to search room availability... | • Search for rooms available by date  
• Search by type of room                                      | 21       |
| 570            | As a vacationer, I want to change my reservation... | • Test objectives...                                                              | 8        |
| 560            | As a vacationer, I want to cancel my reservation... | • Test objectives...                                                              | 8        |

#### Sprint Plan (Stories with Tasks)

<table>
<thead>
<tr>
<th>Test Objectives</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Review</td>
<td>4</td>
</tr>
<tr>
<td>Write Tests</td>
<td>8</td>
</tr>
<tr>
<td>Code</td>
<td>24</td>
</tr>
<tr>
<td>Automate Test</td>
<td>8</td>
</tr>
</tbody>
</table>

#### Daily Commitment

- Daily Stand Up

### Daily Commitment

- Design Review
- Write Tests
- Code
- Automate Test

### Definition of Done

- Designed
- Design Review
- Refactored
- Coded
- Code Review
- Unit tested, functional tested, integration tested,…
- User/Stakeholder Tested
- Documented
Scrum Board in Action!

Check out: http://www.mountaingoatsoftware.com/scrum/task-boards

<table>
<thead>
<tr>
<th>Story</th>
<th>To Do</th>
<th>In Process</th>
<th>To Verify</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a user, I... 8 points</td>
<td>Code the... 9</td>
<td>Test the... 8</td>
<td>Code the... DC 4</td>
<td>Test the... SC 6</td>
</tr>
<tr>
<td></td>
<td>Code the... 2</td>
<td>Code the... 8</td>
<td>Test the... SC 8</td>
<td>Code the... DC 8</td>
</tr>
<tr>
<td></td>
<td>Test the... 8</td>
<td>Test the... 4</td>
<td>Test the... SC 6</td>
<td>Test the... SC 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Story</th>
<th>To Do</th>
<th>In Process</th>
<th>To Verify</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a user, I... 5 points</td>
<td>Code the... 8</td>
<td>Test the... 8</td>
<td>Code the... DC 8</td>
<td>Test the... SC 6</td>
</tr>
<tr>
<td></td>
<td>Code the... 4</td>
<td>Code the... 6</td>
<td>Test the... SC 6</td>
<td>Test the... SC 6</td>
</tr>
</tbody>
</table>
Team Capacity

- Capacity is the team members’ available hours to work in a Sprint
- Revisited each Sprint
- Compare planned task hours to capacity hours

Example for a two-week Sprint

<table>
<thead>
<tr>
<th>Team Member</th>
<th>Hours per day</th>
<th>Total hours in the Sprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>Scott</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>Chris</td>
<td>8</td>
<td>80</td>
</tr>
<tr>
<td>Andy</td>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>Cindy</td>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>Mike</td>
<td>8</td>
<td>80</td>
</tr>
<tr>
<td>TEAM TOTAL</td>
<td>40</td>
<td>400</td>
</tr>
</tbody>
</table>
Agenda

- Iteration 1 – Introduction and Agile Overview
- Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
- Iteration 3 – Release Planning
- Iteration 4 – Writing User Stories
- Iteration 5 – Estimating User Stories
- Iteration 6 – Sprint Planning
- **Iteration 7 – Sprint Execution**
  - Executing the Tasks to get the user stories “DONE”
  - Managing the Sprint Backlog, The Daily Scrum, and Agile Testing
- Iteration 8 – Sprint Demo, Retrospective, and Delivery
- Iteration 9 – Agile Trends and Resources
High Level Agile Stages

- Vision
- Customer Needs
- Product Roadmap
- Release Planning
- Sprint Planning
- Sprint Execution
- Sprint Demo and Retrospective
- Delivery
- Product Backlog
Managing the Sprint Backlog

- Any team member can add, delete or change the Sprint backlog
- Individuals sign up for work of their own choosing
- Estimated work remaining is updated daily
- Work for the Sprint emerges
- If work is unclear, define a Sprint backlog item with a larger amount of time and break it down later
- Update work remaining as more becomes known
- The team collectively is responsible for delivering on their commitments
Sprint Execution: Design, Build, Test

- Unit Testing/Component Testing
- Continuous Integration
- Test Automation
- Peer/Code Reviews

Fixed time frame

The Daily Standup

**SCRUM Daily Standup**

1. What did you do since the last stand up?
2. What will you do today? (Commitment)
3. Is anything in your way?

**SCRUM of SCRUMS – Standup**

(usually two or three times per week)

1. What has your team done since we last met?
2. What will your team do before we meet next?
3. What’s in your team’s way?
4. What are you about to put in another team’s way?

**Parameters**

- Daily
- Attendance required and critical
- 15-minutes
- Stand-up
- Not for problem solving
During the Daily Stand-up team members use the Scrum Board to discuss tasks being worked.
<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### A Team’s Sprint Burndown

#### User Story: As a vacationer, I want to search room availability...

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Tasks</th>
<th>Owner</th>
<th>Status</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Design Review</td>
<td>Scott</td>
<td>Completed</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Install baseline</td>
<td>Bill</td>
<td>Completed</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ICD updates</td>
<td>Scott</td>
<td>Completed</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Acquire test data</td>
<td>Bill</td>
<td>Completed</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Code</td>
<td>Scott</td>
<td>Completed</td>
<td>24</td>
<td>20</td>
<td>16</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Develop tests</td>
<td>Scott</td>
<td>Completed</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Run Tests</td>
<td>Scott</td>
<td>Completed</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

#### The Team

- Manages its work and progress
- Meets daily to discuss progress and commit to plan

#### Managed by the Team
Agile Testing

Test Early  Test Often  Automate
"The error [is] typically 100 times more expensive to correct in the maintenance phase than in the requirements phase."  
*Software Engineering Economics, Barry Boehm*

### The Cost of Change

- **Waterfall**
  - As changes are introduced later in the process, the cost of change increases exponentially.
- **Agile**
  - Changes are made throughout the development cycle, keeping the cost of change relatively flat.

**What if the Cost of Change could be relatively flat? What would that look like?**

Agile attempts to change the cost of change equation.  
*Extreme Programming Explained: Embrace Change, Kent Beck*

- With Agile, it's okay to make changes
- Change is unavoidable
- Agile embraces change
Theory and Practice

• Agile testing is about the people (team and users) and communication
  – Lengthy test documents are often incomplete, out-of-date, ambiguous
  – Test results should be big, public, easy-to-read charts

• Testers are embedded with coders
  – The reasons? Speed, accountability, transparency, transfer of knowledge

• In many cases, the developer is the tester

• Testing is not a phase

• Automated testing is a must

Early, Often, Automate
Levels of Test Automation

www.agilletester.ca
Agenda

- Iteration 1 – Introduction and Agile Overview
- Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
- Iteration 3 – Release Planning
- Iteration 4 – Writing User Stories
- Iteration 5 – Estimating User Stories
- Iteration 6 – Sprint Planning
- Iteration 7 – Sprint Execution
- Iteration 8 – Sprint Demo, Retrospective, and Delivery
  - Reviewing the completed work during the sprint (inspect and adapt)
  - What went well and what could be improved (sprint and releases)
  - Every sprint is potentially releasable
- Iteration 9 – Agile Trends and Resources
High Level Agile Stages

- Vision Customer Needs
- Product Roadmap
- Release Planning
- Sprint Planning
- Sprint Execution
- Sprint Demo and Retrospective
- Delivery

Product Backlog
The Sprint Review

- Demonstrates new functionality
- Transparency and information sharing
- Informal
- Time-boxed
- Everyone invited
- What has been tested and what stories are accepted
- Revisit the backlog
- Update Metrics
Sprint Retrospective

- Take a look at what is and is not working well
- Time-boxed
- Done after every Sprint
- Facilitated by the Scrum Master
- Focus is on process improvement
- Whole team participates
  - ScrumMaster
  - Product owner
  - Team
  - Consider customers and others

Ways to focus the discussion

Goal we want to accomplish

Actions and Priorities

1. 2. 3.

Worked well

Could be improved
A project team’s burndown
(team of teams)

- Based on story points planned
- Updated and reviewed each Sprint
- As stories are accepted and tests passed, requirement progress is updated

A team’s product burndown
Planning the Next Sprint

• The backlog of user stories may have changed
  – Different priorities
  – New user stories

• Customer/Client manager is present to discuss the next set of user stories

• Team reviews estimates in the release or estimates new user stories

• Team chooses user stories that can be done in this next Sprint
  – We have new data about the team velocity

• Team commits to the user stories for the next Sprint

• Team does task breakdown for the chosen user stories
End of the Release

- Similar to end of Sprint – every Sprint is potentially releasable
- Hardening Sprint
- Stories demonstrated and accepted
- Version description document updated
- Many disciplined agile teams have a parallel testing effort during construction Sprints where defects are found and feed back into the process. In addition, the working software becomes a working “system”
- QA testing or similar
Agenda

• Iteration 1 – Introduction and Agile Overview
• Iteration 2 – Team Structure, Vision and Product Roadmap, Release 0
• Iteration 3 – Release Planning
• Iteration 4 – Writing User Stories
• Iteration 5 – Estimating User Stories
• Iteration 6 – Sprint Planning
• Iteration 7 – Sprint Execution
• Iteration 8 – Sprint Demo, Retrospective, and Delivery

• Iteration 9 – Agile Trends and Resources
  – Kanban
  – Agile implementation impediments
Quick Introduction to Kanban Method

• Three Core Principles
  – Start with what you do now
  – Agree to pursue incremental evolutionary change
  – Initially, respect current roles, job titles and responsibilities

• Five Core Practices
  – Visualize
  – Limit Work In Progress (WIP)
  – Manage Flow
  – Make Process Policies Explicit
  – Improve Collaboratively

Our Kanban Board

• Process for organizing maintenance work (patches)

• During sprint planning, we reserve time for Kanban

• Benefits
  – Helps us to manage the unpredictable nature of support work
  – Key information is visible to everyone
  – The team coordinates and updates progress

<table>
<thead>
<tr>
<th>Next</th>
<th>In Progress</th>
<th>Test</th>
<th>Acceptance</th>
<th>Complete!</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Definition of Done
- Code complete & checked-in
- Integrated & tested by developer
- Patch file created

DoD
- Patched installed
- Tested by Support team

DoD
- Customer accepted

Done-done!
“Feel Good” column!

• Work items flow across the board from left to right
• Tasks are part of a work item, the stuff that needs to be done
• Tasks do not flow across the board
Agile Implementation Impediments

- Customer or management not supportive
- Contractual challenges
- Lack of Product Owners
- Lack of experienced or trained Scrum Masters
- Team dynamics cannot support agile
  - Teams too small (eg. 2 or fewer FTEs)
  - Team cannot self manage
  - Team not interested in self-management
Leading Causes of Failed Agile Projects

Find someone to help you through the transition

Tek Talk: Workshop Retrospective

• Reflect on today’s session.
  – What worked well
  – Suggestions for improvement

• What are your next steps?
  – What can you do to make a difference and lead change in your team, program, or organization?

Time: 5 minutes
Final CheckPoint

• Agile is about creating an adaptive organization that is able to respond to the changing needs of customers and technologies
• Agile is not just about software development
• Agile practices affect the entire organization
• There are several Agile methods under the umbrella of “Agile Practices”
• Agile development emphasizes the need for ongoing iterative development with completed, demonstrable functionality at the end of every Sprint
• Agile methods emphasize the need for team and customer collaboration
THE VALUE OF PERFORMANCE.

NORTHROP GRUMMAN
Special Acknowledgments and References

• Many of the ideas in this presentation originated from:
  – Ken Schwaber and Mike Cohn
  – Other contributions/research are noted throughout the presentation
  – My experiences with the many programs and projects across Northrop Grumman

• References and Recommended Agile Reading List
  – *Adaptive Enterprise* by Steven Haeckel
  – *Agile and Iterative Development: A Manager's Guide* by Craig Larman
  – *Agile Estimating and Planning* by Mike Cohn
  – *Succeeding with Agile* by Mike Cohn
  – *Agile Project Management with Scrum* by Ken Schwaber
  – *Agile Testing by Lisa Crispin* and Janet Gregory
  – *Scrum and The Enterprise* by Ken Schwaber
  – *Agile Retrospectives: Making Good Teams Great* by Esther Derby
  – Weekly articles at www.scrumalliance.org
  – www.mountaingoatsoftware.com by Mike Cohn
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint</td>
<td>Fixed time-box in which development occurs also referred to as an “iteration”</td>
</tr>
<tr>
<td>Product Backlog</td>
<td>List of prioritized Requirements/User Stories</td>
</tr>
<tr>
<td>Product Burn Down Chart</td>
<td>Progress for the release; Focuses on the remaining user story points for the given timeframe</td>
</tr>
<tr>
<td>Product Owner</td>
<td>Owns the product backlog, assigns priority to user stories based on customer input</td>
</tr>
<tr>
<td>Release</td>
<td>Usually a 2 – 9 month timeframe; formal committed delivery of product</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>Helps the Agile team through the process and removes impediments</td>
</tr>
<tr>
<td>The Team</td>
<td>Cross functional team</td>
</tr>
<tr>
<td>User Story</td>
<td>Similar to a requirement “As a <strong>user</strong> I want <strong>what</strong> so that <strong>purpose</strong>”</td>
</tr>
</tbody>
</table>