TRAINING AND EDUCATION SERVICES

Agile Team Training Camp
Agile Certified Professional
Participant Guide

TEKsystems
Education Services

TEKsystems
Our people make IT possible

TEKsystems Learning Solutions

presents

Agile Team Training Camp - Agile Certified Professional

Participant Guide

20750 Civic Center Drive
Suite 510, Oakland Commons II
Southfield, MI 48076

AG75-SG-02_20170505

Contents

Facilities and Logistics

Please Turn Off Phone and Computers

Restrooms

Emergency Exits

Sign-in

Breaks and Lunch

Participant Manual

Q & A

Please set your phones to vibrate!

Course Objectives

Upon completion of this course, participants will:

- Explain the key components and principles of Agile and the Lean, eXtreme Programming, Kanban, and Scrum Agile approaches.
- Build a model for Team success based on shared learning within a common context.
- Communicate successfully with Team members and customers.
- Capture the Vision for your product or service.
- Build a Product Roadmap.
- Conduct Release Planning in Agile.
- Build, estimate, and refine a Product Backlog.
- Plan and conduct a Sprint and hold Daily Scrums.
- Adapt the Agile process to reflect continuous improvement.
- Identify and alleviate common impediments to Agile success.

- In addition to the objectives you identified in taking this course, these are the ones we expect to achieve based on the material provided.

Team Training

- This course is intended to give new Agile Teams a start in the right direction and to help established Agile Teams improve their process.
- If the Team is using Scrum, the Product Owner and Scrum Master will be in attendance so everyone can learn about and practice their roles. If the Team is using a hybrid or different approach, the instructor or coach will make the appropriate adjustments.
- Whatever your current scenario, this class is geared toward experimentation, so ask questions, try things out during the simulations, and participate so your Team can make the best choices for your environment.

- This course is intended to give new Agile Teams a start in the right direction or to help level-set Agile Teams who are in flight, but need an opportunity to improve the Team's process.
- Ideally, a Product Owner or Scrum Master would be attending with the Team, so everyone can practice in their roles, but some Teams are not adopting Scrum—they may be working with a more hybrid approach.
- Whatever your current scenario, this class is intended for you to experiment, so ask questions, try things out during the simulations, and participate so your Team can make the best choices for your environment.

Course Input

- This course is designed as an immersion session for Teams at varying stages of an Agile adoption.
- You have been asked to bring actual work to the course that can be used to gain a thorough understanding of the Agile framework.
- This work can be a Vision for a new Product or Service, a Product Backlog, a Requirements Document, User Stories, or Product Backlog Items.
- Simulations will be based on Scrum, but we will discuss influences from other methods.
- If you have attended the Agile Essentials class or have experience with Agile, the background information and/or The Agile Manifesto will be a review, as well as a way to level-set the values and principles with your teammates.

- This course is designed as an immersion session for Teams at varying stages of an Agile adoption.
- You have been asked to bring actual work to the course to use to gain a thorough understanding of the Agile framework.
- This work can be a Vision for a new Product or Service, a Product Backlog, a Requirements Document, User Stories, or Product Backlog Items.
- Simulations will be primarily based on Scrum, but we will discuss influences from other methods.
- If you did attend Agile Essentials or have some experience with Agile, the background information and/or The Agile Manifesto will be review for you and a way to level-set the values and principles with your teammates.

Course Protocols

Approaches to Learning:

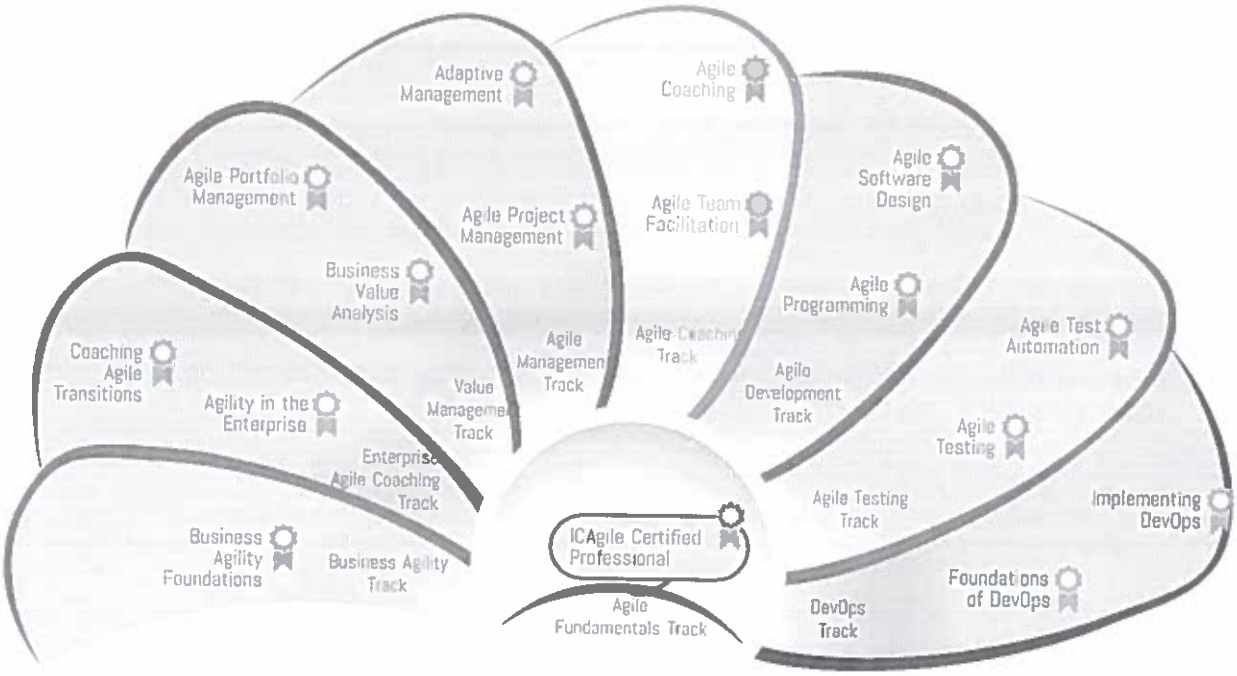
- Sharing, listening, simulating, and doing.
- Respect each other—one conversation at a time.
- The goal is understanding vs. slide coverage.
- Use Backlog for future discussions.
- Respect Timeboxes.

- The success of this class is highly dependent on your participation.
- The goal of this class is for you to achieve true understanding—to have something you can take back and immediately put into practice.

Getting Certified

To get certified:

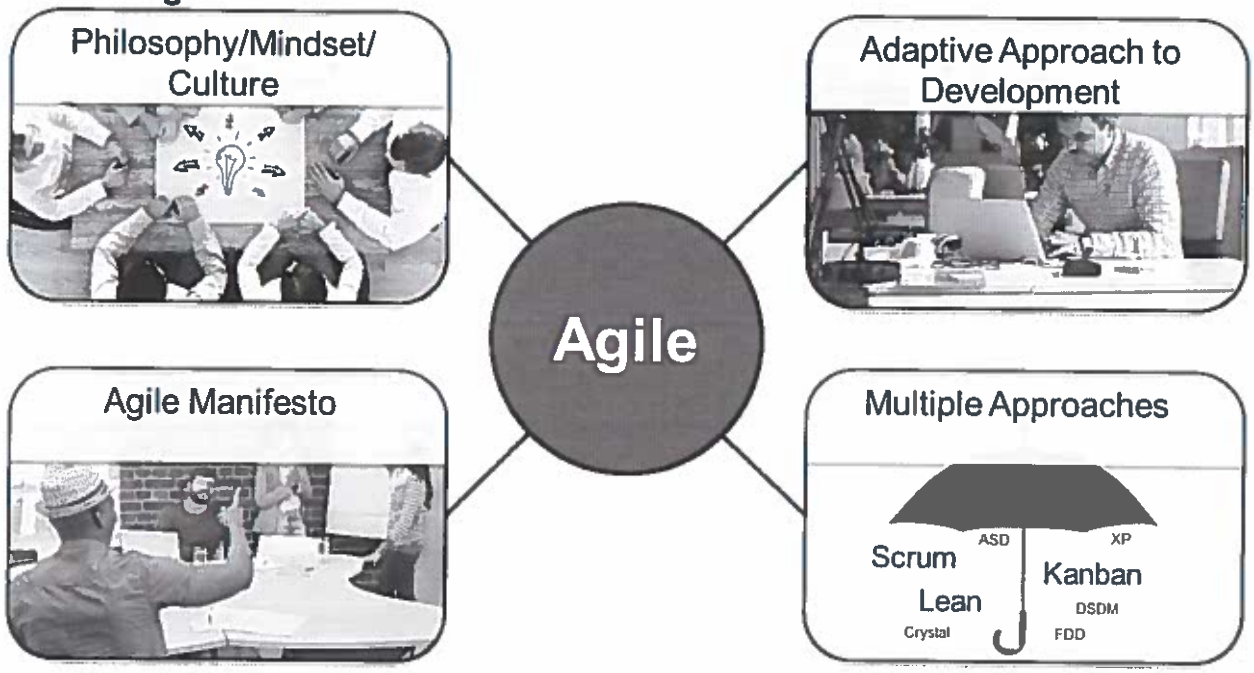
- Attend and participate in class.
- Complete module assessments.



- This is a certification course for ICAgile Certified Professional. More information regarding ICAgile and the related certifications can be found at http://icagile.com.
- The minimum requirements for certification are attendance and participation in class and completing the end-of-module assessments.
- After class, your instructor will submit your name and email address to the IC Agile System.
- You will receive an email with a three question survey. Please check your spam and junk mail folders if you don't receive the email.
- When you respond to the survey, you will receive a link to download your certificate.

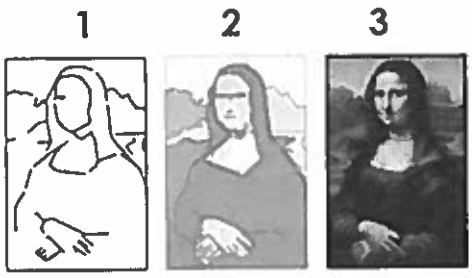
What is Agile?

What is Agile?



- Traditional software or product development methodologies tend to be heavily documented and prescriptive—even providing you templates to use in some cases. Agile is not a true methodology in the traditional sense. It is more of a philosophy or mindset—an adaptive approach to product or software development.
- There is no heavily documented body of knowledge to refer to or set of templates to apply. Instead, the values and principles presented in The Agile Manifesto serve as guidelines for Agile. There is no "one" or "the" Agile method. Any approach or method that puts the Agile values and principles into practice is considered Agile. For example, Scrum, Kanban, Lean, and eXtreme Programming are all Agile approaches.

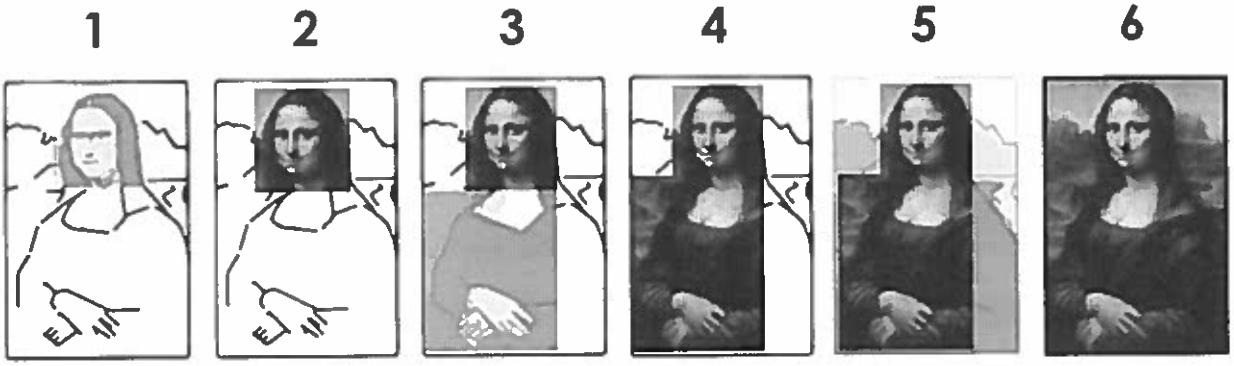
Review – Agile Processes: Iterative and Incremental



Iterative

Incremental

Source: http://jpattonassociates.com/dont_know_what_i_want/



Iterative and Incremental

Source: http://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa

- Regardless of the Agile method, all are iterative, incremental, and evolutionary.
- They follow a similar process to the Plan-Do-Check-Act cycle of Business Improvement.
- This is in contrast to traditional, or more "waterfall" methods, where we do not see the finished product until the end of the cycle.
- The iterative, incremental, evolutionary nature of agility allows for "course corrections" in real-time, as feedback is invited and applied, and Teams inspect and adapt as necessary.
- This means that there are many opportunities to see what is being created along the way.

The 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.

Source: http://agilemanifesto.org/

- The Agile Manifesto, created in 2001, is the glue or basis for all the various approaches (like Scrum or Kanban) that are considered Agile.
- Please note the specific language used in the manifesto's value statements. The manifesto does not say we should never use process, tools, documentation, contracts, or plans. It simply says that we value our conversations and our interactions with customers, people on the Teams, etc., first. We value talking with each other about changes rather than just adhering to a plan when there is a change in our market or industry that we need to respond to.

T.D.D - Test Driven Development.

Review: Twelve Principles of Agile Software

1 Continuous Delivery	7 Working Software
2 Harness Change	8 Sustainable Pace
3 Frequent Delivery	9 Technical Expertise
4 Business and Dev Work Together	10 Simplicity
5 Motivated Individuals	11 Self-Organizing Teams
6 Face-to-Face Conversations	12 Tune and Adjust

Source: http://agilemanifesto.org/principles.html

Deep Work
-book, Cal Newton

The Agile Manifesto has 12 underlying principles.

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

This page intentionally left blank.

Exercise: Principles Bumper Sticker

Overview: Think about some Agile Manifesto principles and sum up their essence by creating a bumper sticker slogan for one or more principles.

Instructions:

1. Self-organize into Teams.
2. Each Team will read The Agile Manifesto.
3. Pick one or two principles that resonate with your Team.
4. If time allows, pick another principle and create another bumper sticker.

Name of Team

Origins of Agile

Agile ideas were being formed even before The Agile Manifesto of 2001.

Before the Manifesto

- Lean Manufacturing
- Boehm's Spiral model
- Rapid development
- Lessons learned
- Plan, Do, Check, Act (PDCA) Cycle
- Rational Unified Process (RUP)
- Dynamic systems development method (DSDM)

After the Manifesto

- Techniques gaining the most attention and use include:
  - Scrum
  - Kanban
  - Lean
  - eXtreme Programming
- New methods continue to evolve:
  - Scaling Agile
  - Agile beyond software
  - New collaboration techniques

- Agile ideas were being formed long before The Agile Manifesto of 2001. Several Agile approaches stem from Japanese manufacturing techniques as old as 1898 and developed in post WWII years. Some of the ideas that led to development of Agile and The Agile Manifesto included:
  - **Lean Manufacturing**: Lean is a systematic method for implementing rapid change through the elimination of waste. Lean's roots trace back to post-WWII at what is now Toyota. Kiichiro Toyoda and Taiichi Ohno identified simple innovations that would make it possible to provide both continuity in process flow, and a wide variety of product offerings.
  - **Boehm's Spiral model:** In the mid 1980s, Barry Boehm introduced the Spiral model with an emphasis on iterative development. You build out more function as you go and increase the amount of work delivered with each iteration.
  - **Rapid development (Rapid Application Development, RAD):** Introduced in the mid 1990's by Steve McConnell, Rapid Development is a collection of best practices to help control development schedules, eliminate waste, and keep projects moving. You would select the processes that would help you. (McConnell)
  - **Lessons learned:** Also called an After Action Review (AAR) or a post mortem, is a review of a project or activity to capture what worked and what didn't and identify areas for improvement.
  - **Plan, Do, Check, Act (PDCA) Cycle:** Also called the Deming wheel, the Deming cycle and Deming's Shewhart cycle, was introduced to William Edwards Deming by his mentor

Walter Shewhart. PDCA is a four step cycle for gaining knowledge and insight to continually improve a product or process.

- **Rational Unified Process (RUP):** RUP is a software development framework focused on managing risk, iterative development, and managing change.
- **Dynamic systems development method (DSDM):** Is a framework originally introduced in 1994 to add support and discipline to RAD. It has continued to be updated and evolved up to the present.

- By the time software leaders met to write The Agile Manifesto in 2001, they were not inventing something new, but simply solidifying what had already emerged.
- After the Manifesto, techniques such as Scrum, Kanban, Lean and XP became much more popular, were improved and applied in many more domains.
- As Agile has become more widely adopted, businesses have discovered that it is good for more than just software. New techniques like scaling Agile to multiple teams, applying Agile to projects and processes that have nothing to do with software, and improving collaboration techniques are evolving.

Assessment: What is Agile?

Questions:

1. True or False? Agile is a prescriptive methodology.
2. True or False? In Agile, we do **NOT** value processes, tools, documentation, and planning.
3. Which of these are Agile principles? Select all that apply.
   A. Face-to-face conversations
   B. Frequent delivery
   C. Gather requirements up front
   D. Working software
   E. Sustainable pace

Why Agile?

Why Agile?

The first thing to understand is WHY you are embarking on Agile.

Agile is not a goal.

(Although some will speak about it as though it is.)

Agile is about enabling business results.

- It is important to note that adopting Agile should not be the goal.
- Agile is a philosophy or a framework to approach solving a complex problem or project.
- Agile is NOT a magic bullet or a silver bullet.
- Adopting Agile will not solve problems that the organization has today.
- Due to the highly transparent nature of Agile, it will very quickly expose impediments and problems that the project or organization already has. It is up to the project Team and/or the leadership in an organization to decide how to address these impediments or problems.

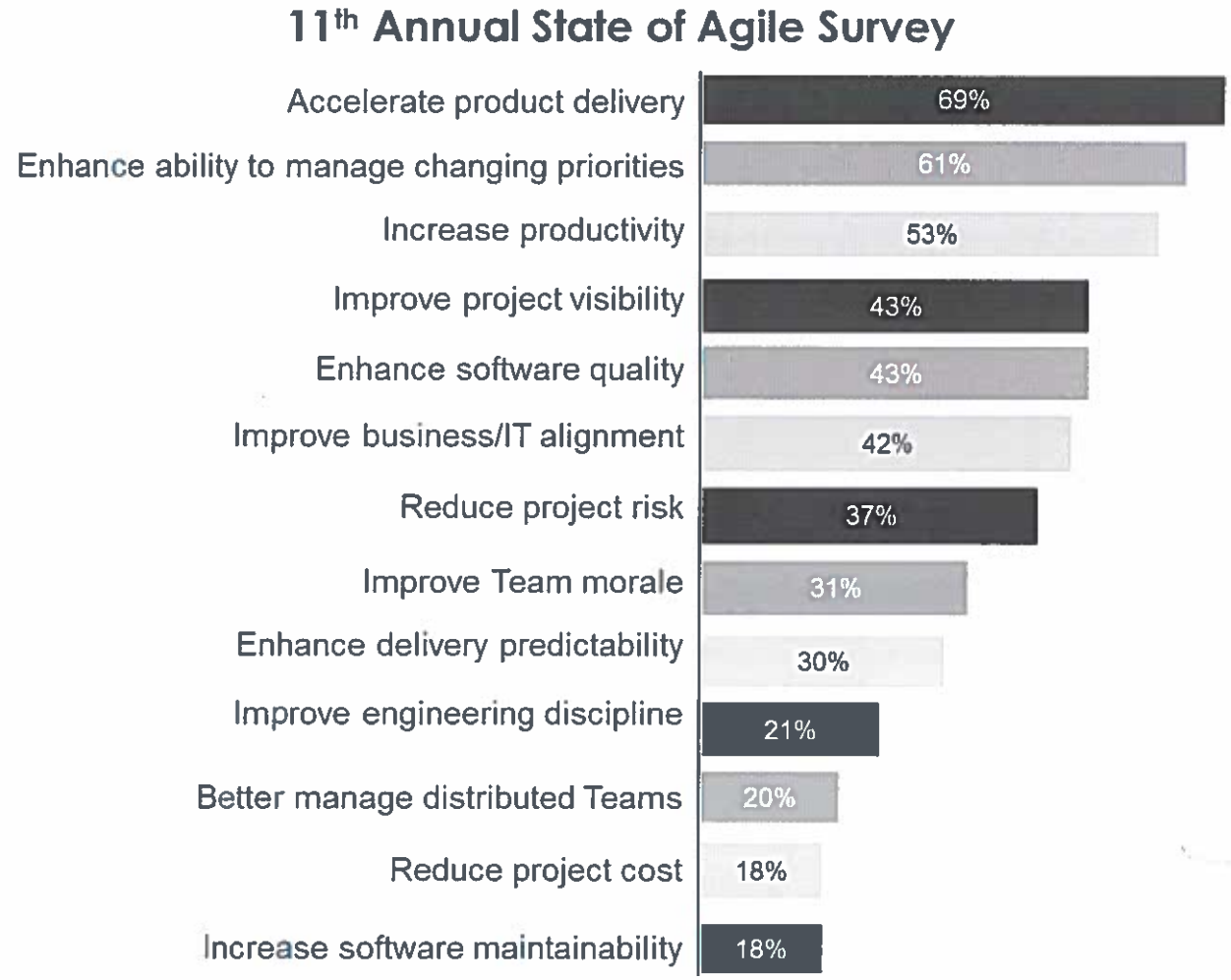
Exercise: Share Your Pain

Question: What problems do you have with your current development process?

Instructions:

1. Self-organize into Teams.
2. Discuss what the Team's goals are and what the company's goals are.
3. Think how you can best use Agile to achieve those goals.
4. Use whiteboard space, flip chart paper, or paper at your tables to record your results.
5. Be ready to share your responses.

Reasons for Adopting Agile

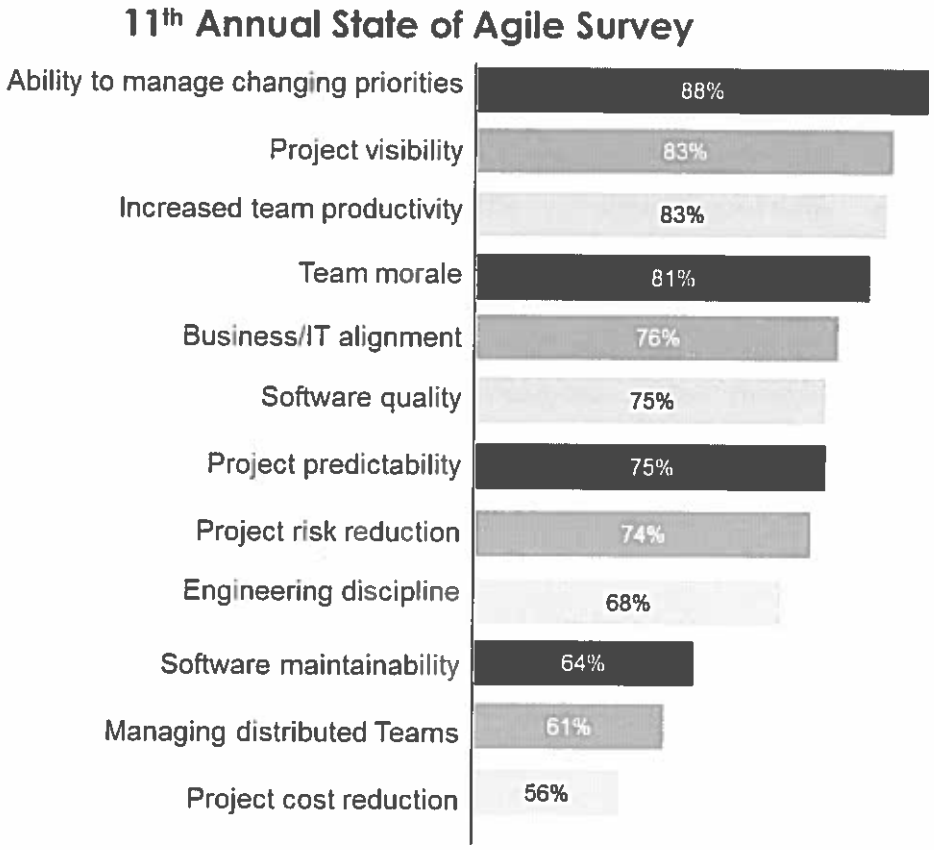


Source: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 (page 8)

VersionOne conducts an annual State of Agile Survey. In their 11th annual survey, the top three reasons cited for adopting Agile were to:

- Accelerate product delivery.
- Enhance ability to manage changing priorities.
- Increase productivity.

Benefits of Agile



VersionOne conducts an annual State of Agile Survey. In their 11th annual survey, they asked participants about the benefits of Agile. Recognizing how Agile can help an organization is important in guaranteeing adoption success.

Discussion: Agile Goals

Questions:

- Why is your Team or organization adopting Agile? (What problem are you trying to solve?)
- What happens if you adopt Agile without a goal?
- What happens if you try to adopt Agile without a commitment to the values and principles?

Instructions:

1. In Teams, discuss the questions on this slide.
2. Use whiteboard space, flip-chart paper, or paper at your tables to record your results.
3. Be ready to share your responses.

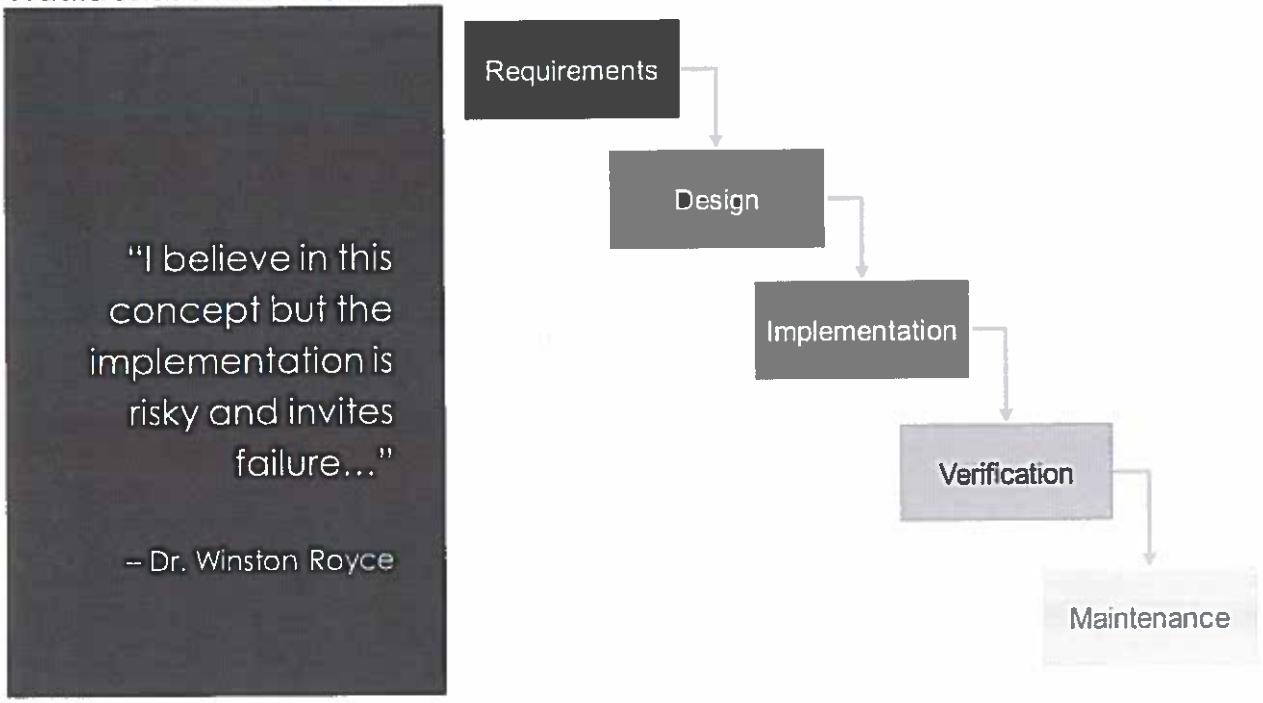
Assessment: Why Agile?

Questions:

1. True or False? Agile is about enabling business results.
2. True or False? Agile will quickly expose impediments and problems the project or organization already has.

How is Agile Different?

Traditional Process: Waterfall



Source: "Managing the Development of Large Software Systems," *Proceedings of IEEE Wescon 26* (August): 1-9

- Most traditional processes, such as waterfall, approach work from Concept through Implementation to Operations and Maintenance in a phased approach: Requirements, design, implementation, verification, and maintenance.
- Each phase is approached in order and generally is departmentalized with a manager of the department providing direction to the Team in addition to a project manager responsible for the overall project.
- Phases do not typically overlap or iterate.
- Lessons-learned ceremonies are usually done at the end of all of the phases, leaving no opportunity to reflect, inspect, adapt, and improve during the process.
- Change is managed tightly with a controlled process that discourages changes during the process.
- The client does not see a finished product until the end of the process.

Discussion: Waterfall

Question: What do you think the issues might be with a traditional waterfall process?

Waterfall: Defined Processes

In a defined process:

- Each phase is clearly defined.
- Repeatability is expected.
- Every piece of work must be completely understood before moving on.

- A *defined process* is a set of clearly defined phases or steps. Each step is tightly integrated with the next step—the output from one is the input to the next. There is an evaluation at the end of each step, but limited feedback within the step.
- The defined process can be started and allowed to run until completion, with the same results (or with little variance) every time given the same input to the process. It requires that every piece of work be completely understood before moving on to the next piece.
- Defined processes work well when a series of steps needs to be repeated without deviation to build an item or a product. However, they invite command and control management—there is no room for independent decision making or empowerment because the steps need to be followed exactly.

Agile: an Empirical Process

An empirical process embraces change as opposed to discouraging it.

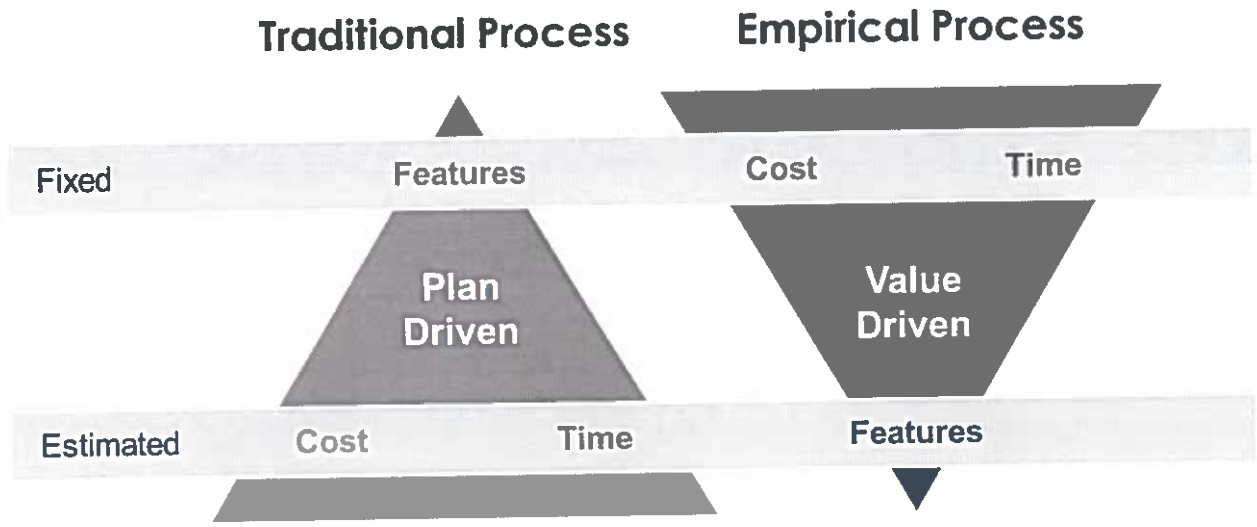
This means that we do not necessarily know a lot of details up front, but rather we learn as we go.

The 3 pillars of any empirical process are:

Transparency **Inspection** **Adaptation**

- In contrast to Waterfall's defined process, Agile approaches are based on an empirical process.
- An *empirical* process is a process that learns from itself over time using data. There are three pillars of any empirical process:
  - Transparency: progress of the work and the Team is visible to everyone.
  - Inspection: there are frequent opportunities to review the current state of the work.
  - Adaptation: the Team makes changes based on the what was learned through transparency and inspection.

The Triple Constraint



Source: Dynamic System Development Method

- This slide is adapted from material on the Dynamic System Development Method (an Agile approach).
- In the traditional process approach:
  - We attempt to define features up front and lock those down.
  - Any changes are managed through control and are discouraged.
  - We estimate costs and time based on only those features defined up front.
- In an empirical process, like any of the Agile approaches:
  - We acknowledge that we usually know our budget (or how much money the customer wants to spend) and we probably have a target delivery date.
  - The only area of flexibility is with the features.
  - If we focus on only highly desired features that will realize the most value for the customer and do not try to include less desired features (which may not be used and thus wasted), we have a better chance of meeting the cost and time constraints.

Discussion: Empirical Process

Question: Why or how would your organization benefit from approaching projects with an empirical process?

Assessment: How is Agile Different?

Questions:

1. True or False? Defined processes invite command and control management.
2. What are the three pillars of an empirical process?
3. True or False? In an empirical process, cost and time are fixed, so the only area of flexibility is the features.

TEKsystems
Our people make IT possible.

Agile Approaches

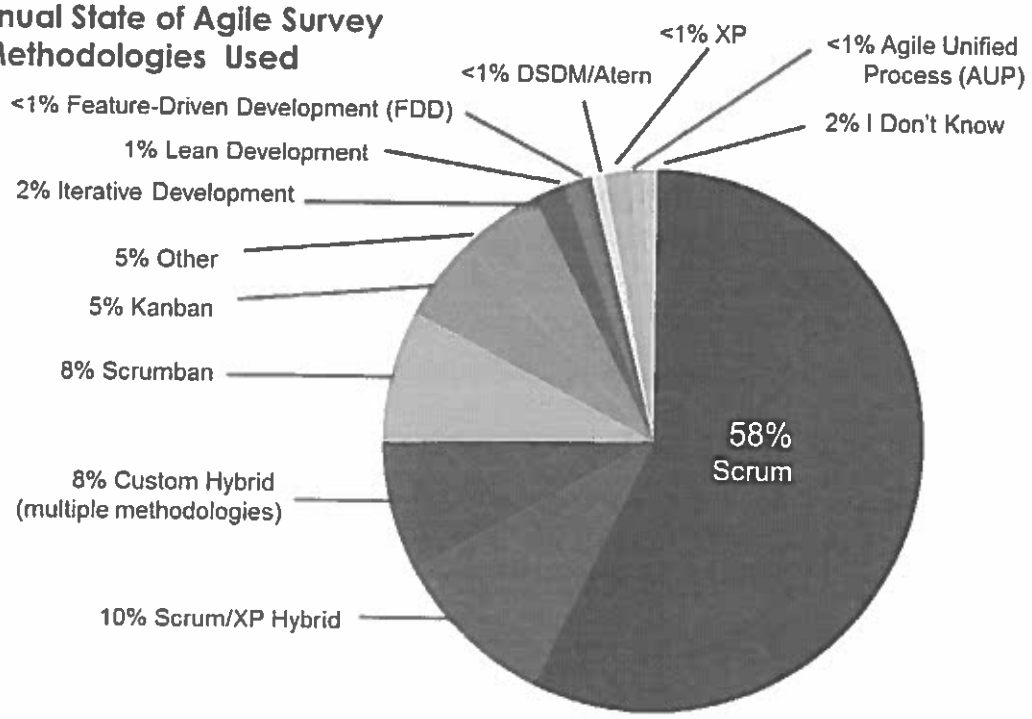
Agile Methods

Many frameworks and approaches fall under the "Agile umbrella."

- The different methods and approaches listed here are all considered "Agile" or fall under the "Agile umbrella."
- All of these methods adhere to the values and principles in The Agile Manifesto.
- We will be discussing the more popular of these methods in this course.

Agile Methodologies in Use

11th Annual State of Agile Survey
Agile Methodologies Used



Source: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 (page 10)

- The survey results are part of VersionOne's 11th Annual State of Agile Survey.
- Although there are many "flavors" of Agile, the most popular approach by far remains Scrum or a Scrum/XP hybrid approach.
- We will cover elements of the most popular methods with a focus on Scrum, due to its widespread adoption and popularity.
- You are encouraged to adopt or use the method, or combination of methods, that will best solve the problem you are facing or the type of work your Team, organization, or company does.

Lean Software Development

Lean Development Principles

Lean development is summarized by seven principles:

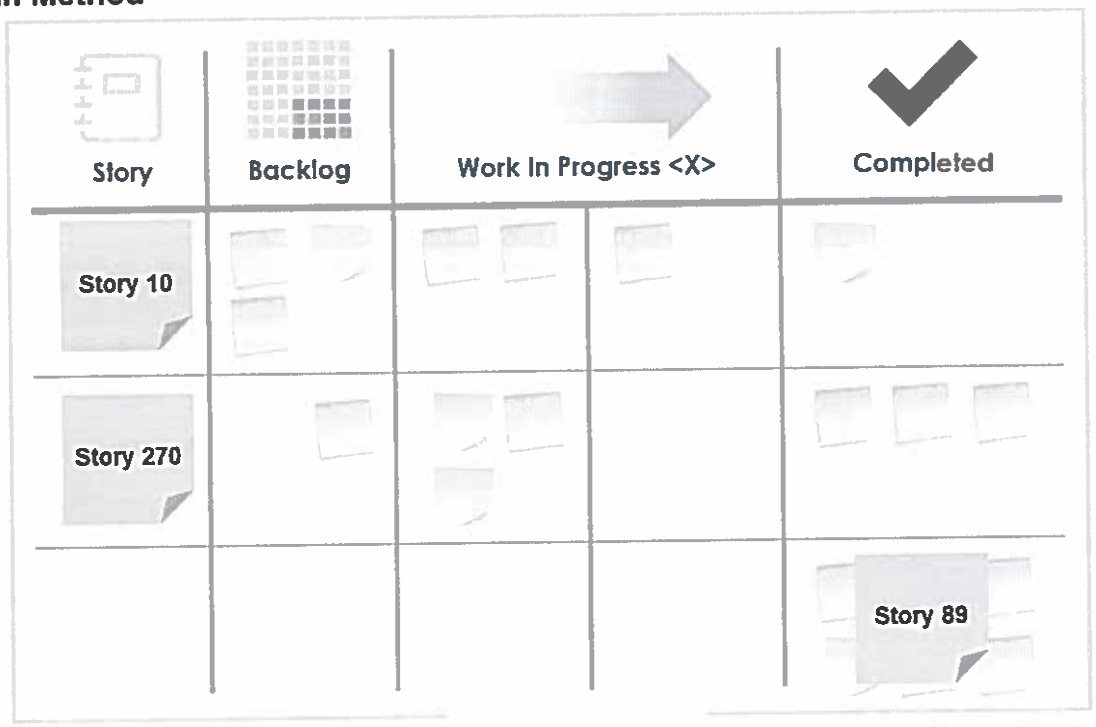
- Eliminate Waste: Unnecessary Code or Functionality
- Amplify Learning
- Decide as Late as Possible ("Last Responsible Moment")
- Deliver as Fast as Possible
- Empower the Whole Team
- Build Integrity In
- See the Whole

- Lean development can be summarized by seven principles, very close in concept to lean manufacturing principles:
  - **Eliminate Waste**: We want to eliminate unnecessary code or functionality. Keep in mind that while some forms of waste are obvious, others are hard to find. Other types of waste to watch out for include: unnecessary code, stating more than can be completed, delay in process, bureaucracy, slow\ineffective communication, defects, or task switching.
  - **Amplify learning**: We want to create knowledge and amplify the learning of the team. Activities to consider would include pairing, code reviews, comments, and sharing.
  - **Decide as late as possible**: When you have a set of options, make the decision as late as possible, especially for things that are impractical to reverse.
  - **Deliver as fast as possible**: Provides a competitive advantage. Quicker feedback. And less chance that there will be changes between what was requested and what gets implemented.
  - **Empower the whole Team**: Respect everyone. Listen attentively, hear opinions, and encourage input.
  - **Build integrity in**: Quality and integrity of the solution should be a focus right from the beginning. You can add in quality at the end.
  - **See the whole**: Look to optimize the entire value stream. By viewing the whole system, you are able to avoid unnecessary optimization.
- Many of the concepts and ideas outlined in Agile processes are rooted in Lean.

- Lean focuses on the customer's desired value and determines the best way to deliver that value while eliminating waste in the process.
- Lean thinking allowed Toyota to shift the focus of the manufacturing engineer from individual machines and their utilization, to the flow of the product through the total process.
- Toyota concluded that it would be possible to obtain low cost, high variety, high quality, and very rapid throughput times to respond to changing customer desires by doing the following:
    - Right-sizing machines for the actual volume needed.
    - Introducing self-monitoring machines to ensure quality.
    - Lining the machines up in process sequence.
    - Pioneering quick setups, so each machine could make small volumes of many part numbers.
    - Having each process step notify the previous step of its current needs for materials, it would be possible to obtain.
- Also, information management could be made much simpler and more accurate.
- The term "Lean Software Development" originated from the title of Tom and Mary Poppendieck's book, *Lean Software Development*. (2003).

Kanban

The Kanban Method



Explicit Policies and Exit Criteria:	Definition of Ready	I.N.V.E.S.T.	WIP limit = 6. Acceptance Criteria met and UAT completed. Definition of Done.	Items remain visible for 14 days then get archived

- The name Kanban originates from Japanese and very roughly translates as "signboard." It promotes a continuous flow of work.
- Like Lean, Kanban is also traced back to Toyota. The need to maintain a high rate of improvement led Toyota to devise the Kanban system.
- Taiichi Ohno produced Kanban to control production between processes, and implemented Just In Time (JIT) manufacturing.
- David J. Anderson is credited with formulating this into an incremental, evolutionary process and systems change for organizations.
- The method uses a pull system that employs work in progress (WIP) limits as its core mechanism.
- Kanban is not an inventory control system. It is a scheduling system that helps determine what to produce, when to produce it, and how much to produce.
- There are no Iterations or Sprints with Kanban. Visibility is given to WIP, and limits are imposed on how much WIP is done at a time.
- Items go out the door when they are "done."

- Kanban does not define a set of roles or process steps. Instead, Kanban starts with the roles and processes you have and stimulates continuous, incremental changes to a system.
- Although Kanban can be effective for software development Teams, many Agile adoptions have this applied to Help Desk, Operations, Infrastructure, and Data Warehouse Teams. In those Teams, it would be unrealistic for someone to wait until the end of a two-week Sprint for their requested functionality—the expectation is that they would have this sooner or as soon as it is "done."

Core Principles and Properties

Kanban is based on these core principles:

- Start with the existing process.
- Agree to pursue incremental and evolutionary change.
- Respect the current process, roles, responsibilities, and titles.
- Encourage acts of leadership at all levels.

And adheres to these properties:

- Visualize the flow.
- Limit Work in Progress (WIP).
- Measure and manage flow.
- Make process policies explicit.
- Improve continuously and collaboratively.

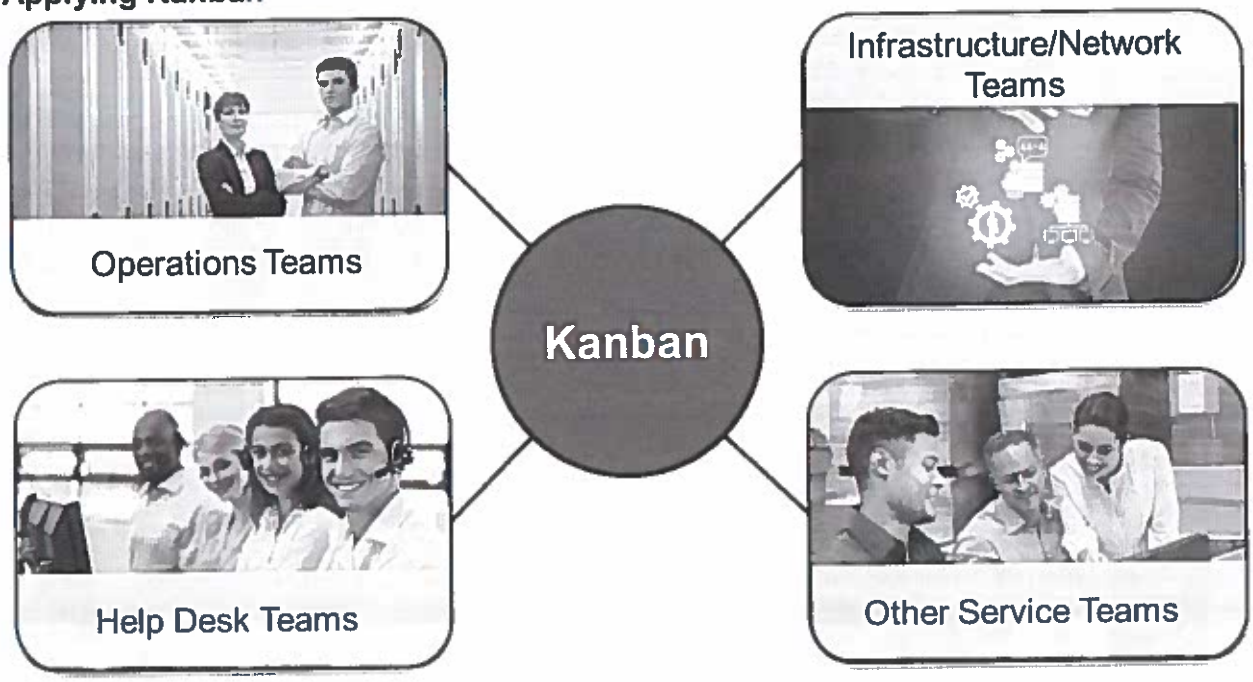
Kanban-Successful Evolutionary Change for your Technology Business, by David Anderson

- Kanban is based on these core principles:
  - **Start with the existing process**: Kanban does not prescribe a process. It starts with your existing process, limiting the amount of initial change.
  - **Agree to pursue incremental and evolutionary change**: Small, incremental changes allow for easy impact discussion and minimize resistance.
  - **Respect the current process, roles, responsibilities, and titles**: Your current process, roles, responsibilities, and titles have value. Kanban does not prohibit or prescribe change.
  - **Encourage acts of leadership at all levels**: Everyone, regardless of formal title, needs to foster a continual improvement mentality.
- To succeed with Kanban, you want to adhere to these properties:

- **Visualize the flow**: A common way to visualize the workflow is to use a card or wall or Kanban board with cards and columns. Teams can have as many columns on their board as is deemed necessary. More columns are added for increased visibility or if there are more steps in the process.
- **Limit Work in Progress (WIP)**: The idea is not to overload the Team, but to limit the WIP to one item at a time. This implies that a pull systems is implemented in parts or all of the workflow.
- **Measure and manage flow**: The flow of work through each state should be monitored, measured, and reported. If an item becomes blocked, a Team member can always pull in the next highest priority item until their blocker is removed.
- **Make process policies explicit**: An explicit understanding makes it possible to move to a more rational, empirical, objective discussion of issues.
- **Improve continuously and collaboratively**: Teams are encouraged to be self-organizing and to work collaboratively. A shared understanding of theories about work, workflow, process, and risk is more likely to result in a shared comprehension of a problem and improvements that can be agreed on by consensus.

- Some Teams find success in combining Kanban with ideas from Scrum, having a Daily Stand Up meeting around their Kanban Board and also holding Retrospectives periodically to identify process improvement. Corey Ladas wrote the popular "Scrumban," which discusses this hybrid approach.

Applying Kanban



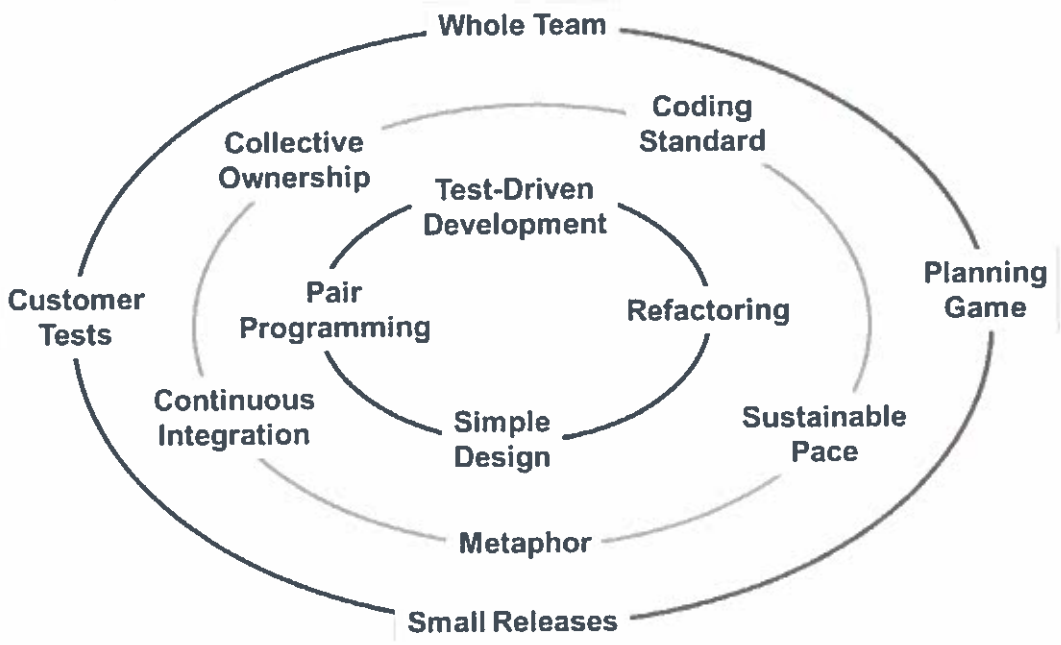
- In order for this to be effective, the organization or Team must agree that this approach is the way to improve the system and then stick to it.
- Let's look at some of those applications outside of software development:
  - Operations Teams: Kanban can be very effective for Teams that handle regular maintenance tasks.
  - Infrastructure/Network Teams: Kanban can be very effective for infrastructure and network Teams where it might not make sense to contain work in Sprints or Iterations.
  - Help Desk Teams: Help Desks can use Kanban to help track and resolve reported user issues.
  - Additional Services Teams: Teams such as Data Warehouse Teams or Teams that handle end-user requests that require a quick turnaround can also benefit from a Kanban approach.

Discussion: Kanban

Questions:

- What aspects of Kanban are interesting to you? Why?
- How does Kanban support the values and principles of The Agile Manifesto?

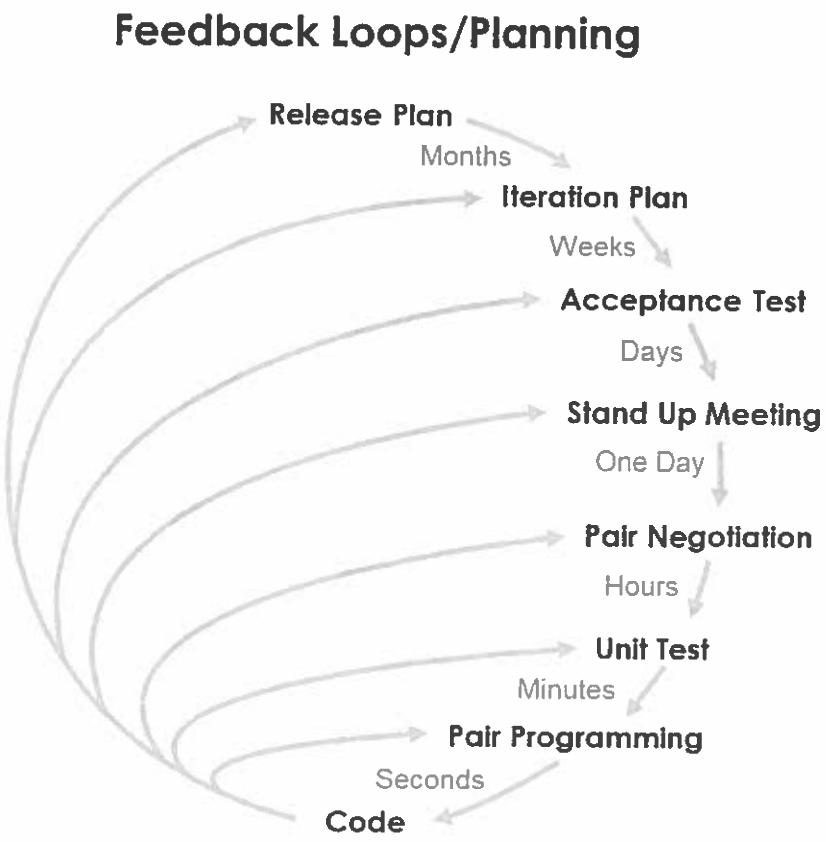
eXtreme Programming (XP)



Source: http://ronjeffries.com/xprog/what-is-extreme-programming/

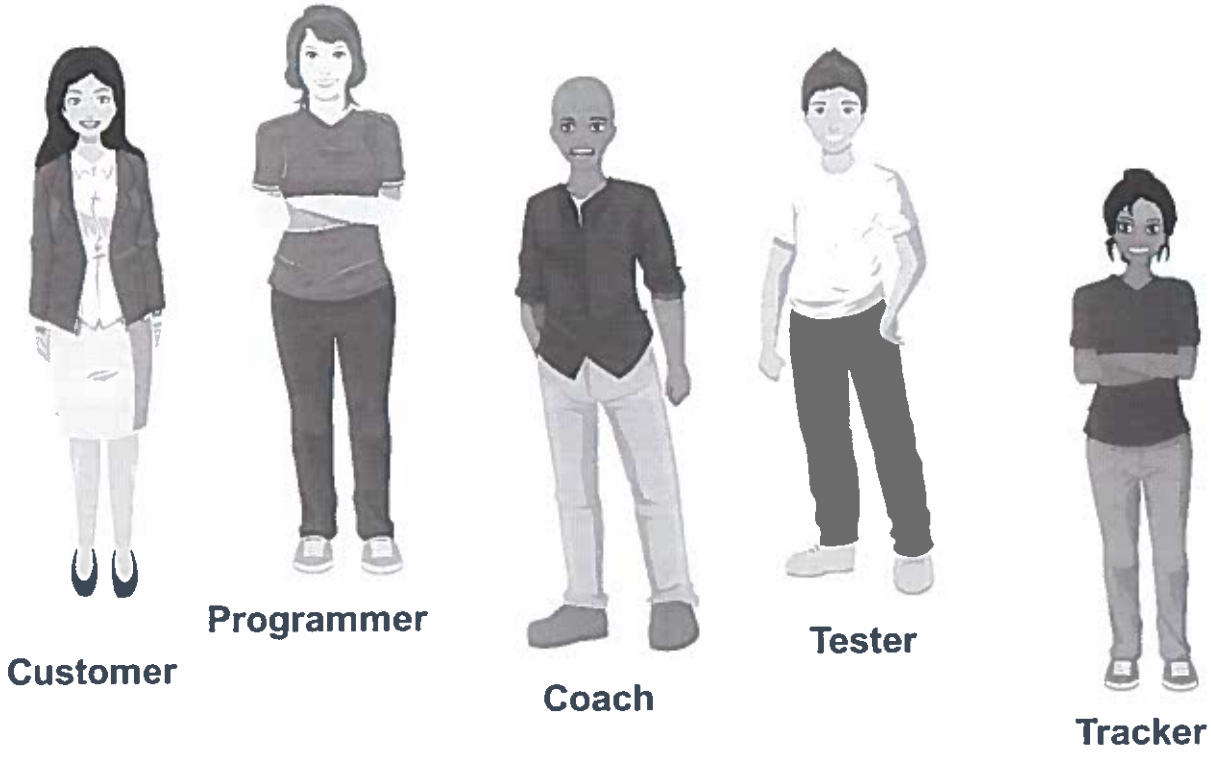
- eXtreme Programming (XP) is another method under the Agile umbrella. It was created by Kent Beck during his work on the Chrysler Comprehensive Compensation System (C3) payroll project. He became the C3 project leader in March 1996 and began to refine the development method used in the project. He wrote a book on the method (*Extreme Programming Explained*, October 1999).
- XP is all about improving software development practices, moving the quality up in the process and being responsive to changing customer requirements. It advocates frequent releases in short development cycles, intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. It relies heavily on feedback and design is a continual process rather than completed upfront. The methodology takes best practices (for the practices listed on the slide) to extreme levels.
- XP Teams sit together in one lab, space, or large room and keep the system integrated and running all the time.
- XP is based on these values:
  - Simplicity.
  - Communication.
  - Feedback.
  - Courage.
  - Respect.

XP Feedback Loops and Planning



- In addition to continual feedback loops, XP has two different levels of planning: Release Planning and Iteration Planning. XP Planning addresses two key questions: What will be accomplished by the due date, and what to do next?
- Release Planning is a practice where the customer presents the desired features to the programmers and the programmers estimate their difficulty. The customer then lays out the plan for the project based on the importance of the features and the estimates.
- Release Planning is formally recognized in which the customer describes the overall functionality, features, etc.
- Iteration Planning gets the Team on board with what they will be working on during the Iteration. The Team determines how many Iterations they will need to achieve the Release. XP Iterations are typically one week to two weeks long at the most.

XP Roles



Most XP Teams include the following roles, which may be shared by members of the Team.

- Customer: The customer is in the driver's seat, defining what will be built and driving the project. The customer represents the business interests and the end user and holds a decision-making position; defining and prioritizing the features.
- Programmer: The programmer(s) are responsible for turning the customer's requirements into working code. They raise issues with customers and estimate their work and then develop, implement, and maintain the code.
- Tester: The tester helps the customer define the acceptance tests and then runs the tests and posts results.
- Coach: The coach guides, teaches, and mentors the Team to ensure the team stays on track and help improve the process.
- Tracker: The Team may have a tracker who keeps track of schedule, progress, and rate of progress. The tracker monitors the Release Plan, iteration plan, and acceptance tests.

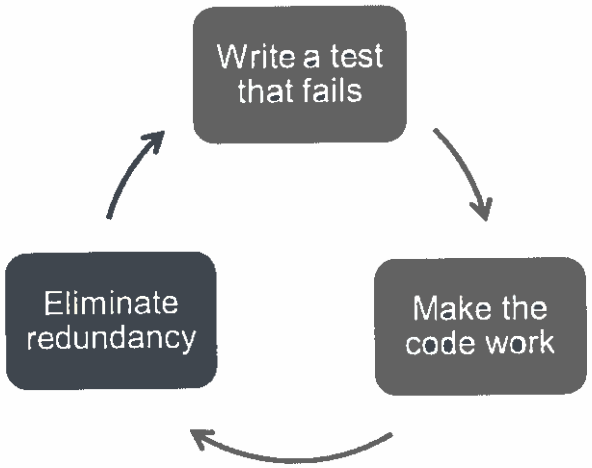
XP Practices: Pair Programming

Pair Programming:

- Technique in which two Team members work together at one computer. One person is typically writing code, while the other is performing real-time code review.
- The people working in the pair can switch off as needed.
- Pairs typically complete work faster than one person assigned to do the same task alone.
- Errors are usually caught earlier, reducing the overall defect rate and improving the quality of the product.

- Pairing can also take place by having one person who has traditionally been a Quality Assurance person, paired with the programmer performing real-time review or acceptance testing.
- Someone who has traditionally worn a Quality Assurance hat can perform the review or assist from a testing perspective if they are able to do so in the pair.
- Discuss with other class participants: What other pairing possibilities can you think of? Do you use pair programming practices in your Teams today?

XP Practices: TDD



Test Driven Development (TDD):

- Is a technique in which a Unit Test is written first.
- Helps us define what the function or requirement should do.
- Involves code that is first written to pass the test and then improved.
- Moves quality up in the process.
- Prevents defects as opposed to hunting for them later.

- The goal in writing test cases first and then writing code to pass is to move quality up in the process; preventing defects, not necessarily waiting until the end of a project to hunt for them.
- If there are Team members who are not comfortable creating the acceptance tests before writing code, who on the Team can assist in this effort?
- The goal is not to set up siloed functions by reinforcing traditional roles.
- The goal is to have the Team working together to move quality up in the process and produce working software.

XP Practices: Refactoring

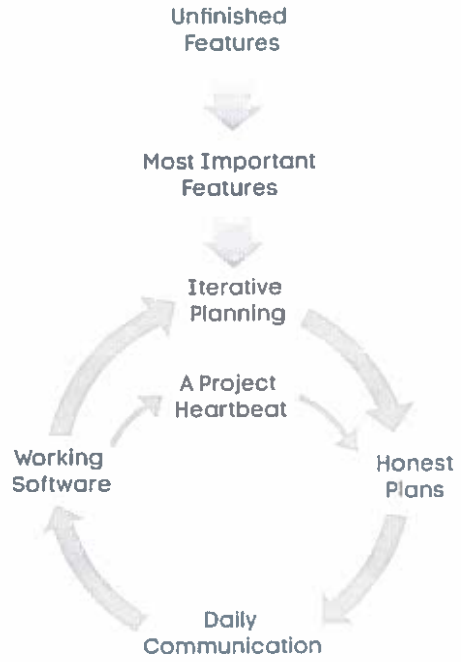
"Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure."

- Martin Fowler, *Refactoring: Improving the Design of Existing Code*

- Disciplined technique for improving or restructuring existing code.
- Typically involves **small changes** that *do not* modify requirements.
- Improves code readability, reduces complexity, and improves maintainability of source code.
- Examples of when to refactor include:
  - Several nested queries were used where one, more efficient query, could have been put into place.
  - Code was written to perform a function where a reusable library was available.

- Refactoring is a programming practice that has to do with the code itself in terms of its ease of being maintained, commented, read, etc.
- Refactoring examples:
  - Perhaps several nested queries were used when one, more efficient query, is known and can be put into place.
  - Maybe code was written to perform a function where a reusable library was available to call.
- Refactoring does not mean rework or a change in requirements.
- An example of what is NOT Refactoring, is if a Product Owner asked for a check box, but now that they see the functionality, they request a radio button instead. Refactoring is more about the process.

XP Practices: Continuous Integration



- Involves frequently integrating new code with existing code to check for errors.
- Moves quality up in the process.
- Helps *prevent* defects as opposed to searching for them later.
- Differs from traditional practices where quality is tested *after* code development.

Source: http://www.extremeprogramming.org

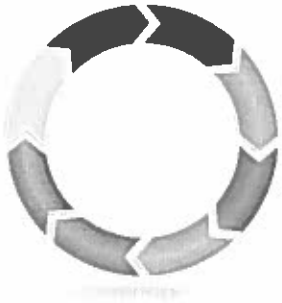
- Continuous Integration is another opportunity to move quality up in the process to prevent defects rather than to find them later in the life cycle.
- Many organizations have built infrastructure and processes around more traditional or waterfall-type processes.
- Moving to a Continuous Integration environment and investing in automated test tools to perform the regression testing needed is an investment, but will yield fewer defects and higher-quality software in the long run.
- Discuss with other class participants: Do you currently have Continuous Integration in place? What would it take for you to achieve this from an infrastructure perspective? A process perspective?

Continuous Delivery (CD)

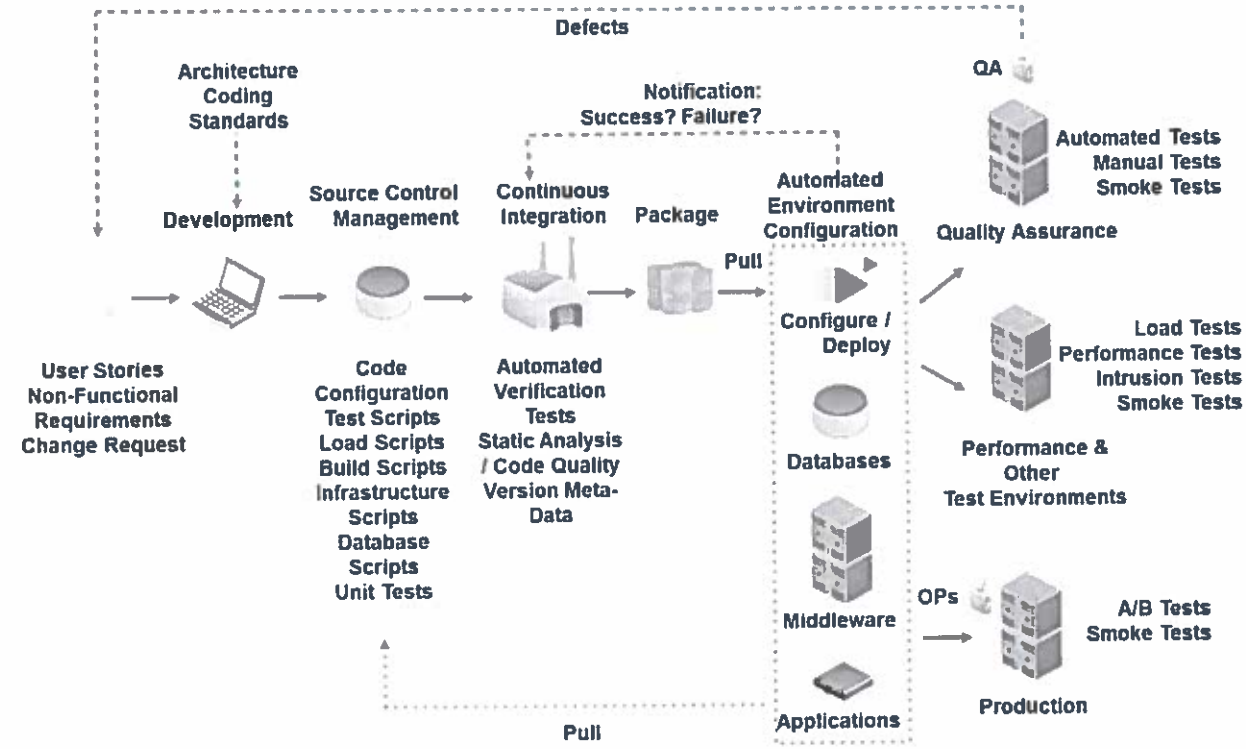
Taking CI one step further:

- **Continuous Delivery (CD)** is a continuation of CI that adds the step of deployment automation.
- Upon a successful build, the executable is deployed/installed either to a UAT or Staging environment.
- Since all changes delivered to the environment are automated, you can have confidence that the application can be deployed to production:
  - This deployment can happen at the push of a button—once the business is ready.
- **Continuous Deployment** is the next step of Continuous Delivery. Each change that passes automated testing gets deployed to production automatically.

- Teams implementing Continuous Delivery (CD) must have a clear understanding that anything checked in can end up in production.
- CD is a natural extension of Continuous Integration (CI), so while it is not part of XP, there is a natural alignment.
- Here is an interesting article discussing the difference between Continuous Delivery and Continuous Deployment:
  - https://puppetlabs.com/blog/continuous-delivery-vs-continuous-deployment-whats-diff



Continuous Delivery Pipeline



- The goal of Continuous Delivery is to enable a constant flow of changes into production via an automated software production line.
- Continuous Delivery pipeline:
  - Breaks down the software delivery process into stages.
  - Each stage is aimed at verifying the quality of new features from a different angle to validate the new functionality and prevent errors from affecting your users.
  - Should provide feedback to the Team and visibility into the flow of changes to everyone involved in delivering the new features.
- Typical CD pipeline will include:
  - Build automation.
  - Continuous Integration.
  - Test automation.
  - Deployment automation.
- An important piece in this pipeline is version control. Who uses version control today? Everyone!
- Using version control is an old-school practice, so why talk about it with DevOps?
- What do you currently put into version control?
  - You version control everything!

- Code (of course!).
- Scripts:
    - Build Images/Build Containers/Configurations/Create Builds.
    - Other artifacts and documents.

Class Discussion: XP

Questions:

- What aspects of XP are interesting to you? Why?
- How does XP support the values and principles of The Agile Manifesto?

Assessment: Agile Approaches

Questions:

1. True or False? Lean focuses on the customer's desired value and determines the best way to deliver that value while eliminating waste in the process.
2. Which Agile approach uses a pull system and WIP limits to manage work?
   A. Lean
   B. Kanban
   C. XP

   Work In Progress
3. Which XP practice involves frequently integrating new code with existing code to check for errors?
   A. Pair Programming
   B. TDD
   C. Refactoring
   D. Continuous Integration

TEKsystems
Our people make IT possible.

Scrum

What is Scrum?

Scrum is:

- A management framework for completing complex projects/products.
- Performed by a cross-functional Scrum Team.
- Iterative and incremental.
- About transparency, inspection, and adaptation.

- In the early 1990s, Ken Schwaber used what would become Scrum at his company, Advanced Development Methods.
- Jeff Sutherland, with John Scumniotales and Jeff McKenna, came up with a similar approach at Easel Corporation, and were the first to refer to it using the single word *Scrum*.
- Scrum is:
  - An innovative, management framework for completing complex projects/products.
  - Supportive of the values and principles in The Agile Manifesto.
  - Performed by a cross-functional Scrum Team. It is a Team approach, not a group approach.
  - Iterative an incremental.
  - About transparency, inspection, and adaptation.
- The word "Scrum" is a Rugby term, representing Team members working together tightly, to move the ball down the field quickly. Rugby play cannot be extensively planned; players have to adjust very quickly in the moment.
- Scrum is based on these five values:
  - Commitment.
  - Focus.
  - Openness.
  - Respect.
  - Courage.
- Scrum is iterative and incremental.
- The three pillars of Scrum are transparency, inspection, and adaptation.

Scrum Flow

This view provides a holistic view of the extended Scrum process. We will be covering each of these areas in detail in this course.

- Vision and Roadmap: These are optional in Scrum, but provide the big picture for the project.
- Release Planning: Details how we will get to the end product.
- Product Backlog: The Product Backlog is an ordered and prioritized list of everything that might be in the product.
- Sprint Planning: The entire Scrum Team gets together to plan the Sprint. The Development Team pulls a small chunk from the top of the ordered Product Backlog and creates a plan (Sprint Backlog) for how to implement those Product Backlog Items (PBIs) during the upcoming Sprint.
- Sprint Backlog: Created by the Development Team during Sprint Planning, the Sprint Backlog is the plan for how to implement the Product Backlog Items (PBIs) during the upcoming Sprint.
- Sprint: An iteration of one to four weeks where the Team completes the work they agreed to during Sprint Planning. The objective out of each Sprint is to deliver working software—even if it is an increment.
- Daily Scrum: A daily event in which the Development Team members check in with each other and synchronize information, raise impediments, and inspect and adapt as necessary. Only Development Team members participate in the Daily Scrum.
- Sprint Review: An opportunity for the Scrum Team to demonstrate the Potentially Releasable Increment created during the Sprint to the key stakeholders. Feedback received is incorporated back into the Product Backlog. The Scrum Team and stakeholders then reorder the Backlog for upcoming Sprints.

- Sprint Retrospective: An inspect and adapt mechanism at the end of each Sprint that allows the Scrum Team to improve their effectiveness, discussing what went well, what didn't, and what they will attempt to improve in the coming Sprints.
- Potentially Shippable Product Increment: Is the end result of each Sprint. A Potentially Shippable Product Increment that functions is still considered valuable—even if we save those increments to be released to production together after a certain number of Sprints (a.k.a., Release).

Typical Scrum Schedule

	WED	THU	FRI	MON	TUE
SPRINT 1	Sprint Planning	Daily Scrum	Daily Scrum Backlog Refinement	Daily Scrum	Daily Scrum
	Daily Scrum	Daily Scrum	Daily Scrum Backlog Refinement	Daily Scrum	Sprint Review Retrospective
SPRINT 2	Sprint Planning	Daily Scrum	Daily Scrum Backlog Refinement	Daily Scrum	Daily Scrum
	Daily Scrum	Daily Scrum	Daily Scrum Backlog Refinement	Daily Scrum	Sprint Review Retrospective

- The slide depicts an overview of Scrum events for a two-week Sprint, currently the most common in the industry.
- Scrum schedules include these events:
  - Sprint Planning.
  - Daily Scrum.
  - Sprint Review.
  - Sprint Retrospective.
- The Sprint schedule should always start with a Sprint Planning event, and end with the Sprint Review, and Sprint Retrospective events.
- The Sprint is not done until the Scrum Team has conducted their events.
- The Daily Scrum is a daily event. It is key that the schedule of this important event be well understood by the Development Team.
- All four prescribed Scrum events are mandatory. Backlog Refinement, while required, is not an event, but a required activity.
- There is no lag time between Sprints, e.g., Sprint 1 and Sprint 2.
- The Scrum schedule never stops. Once you start Sprinting, you just keep going!

Discussion Question

Question:

- What aspects of Scrum are interesting to you? Why?
- How does Scrum support the values and principles of The Agile Manifesto?

Assessment: Scrum

Questions:

1. True or False? Scrum is a management framework for completing complex projects and products.
2. Which of the following items is optional in Scrum?
   A. Product Backlog
   B. Vision and Roadmap
   C. Daily Scrum
   D. Sprint Retrospective
3. Which event kicks off the Sprint?
   A. Sprint Planning
   B. Daily Scrum
   C. Sprint Review
   D. Backlog Refinement

The Players

Scrum Roles

Product Owner

Scrum Master

Development Team

- The Scrum Team consists of a Product Owner, a Scrum Master, and the Development Team:
  - Product Owner: Is one person responsible for maximizing the value of the product and the work of the Development Team.
  - Scrum Master: Is the servant-leader responsible for ensuring the Scrum Team adheres to Scrum theory, practices, and rules and that the organization understands Scrum.
  - Development Team: Do the work required to deliver a Potentially Shippable Increment at the end of the each Sprint. Generally, there are three to nine members on the team. The Development Team is dedicated to one project at a time.
- Scrum Teams are self-organizing and cross-functional; they are not locked into the boundaries of traditional roles.
- Those formerly known as Business Analysts may find themselves executing acceptance tests, or those formerly known as Developers may find themselves writing test cases before coding to ensure that the Acceptance Criteria are met.
- Cross-functionality is not something that will happen overnight.
- Discuss with the other class participants: What are ways you can think beyond limits of traditional roles to begin moving toward cross-functionality?

The Product Owner

- Shares the Vision with the Scrum Team.
- Orders Product Backlog Items (PBIs) (features) based on business value, developing the Roadmap and Release Plan, and maintaining them as necessary.
- Creates and "refines" the Product Backlog with the Dev Team, as an ongoing activity to provide the right level of detail at the right time.
- Collaborates with the Scrum Team at all times.
- Lead facilitator at Sprint Reviews and Release Planning (optional practice).
- Participates in all Scrum events.
- Accepts or rejects PBIs during the Sprint.

- The Product Owner role is one of the most critical roles in Scrum.
- Without this person communicating the Vision and providing the business priority for features, the Scrum Team does not have a direction for focusing on their work.
- We discuss the Product Owner role briefly in this course so that Team members have a basic understanding of the role and what is expected from the person filling this role.
- This class is not a deep dive specifically into this role.
- For more information on the Certified Scrum Product Owner training, please contact TEKsystems Education Services.
- If your Team's Product Owner is in attendance with you for this course, it will help the simulation exercises in this course be as "real world" as possible.

The Product Owner (Cont.)

Product Owner:

- Has the authority to make decisions, is decisive, and is willing to say no.
- Has the knowledge of the market, business domain, clients, and users.
- Has the availability to communicate effectively with the Dev Team and stakeholders.
- ***The Product Owner is one person***, not a committee or Team.
- Is responsible for optimizing the value of the product via the Product Backlog.
- Leads all Sprint Review discussions about the Product Backlog with key stakeholders.

- One of the most important characteristics of the Product Owner is that they are empowered to make decisions.
- The Product Owner has ownership of the product, including financial responsibility and veto power—the ability to say no.

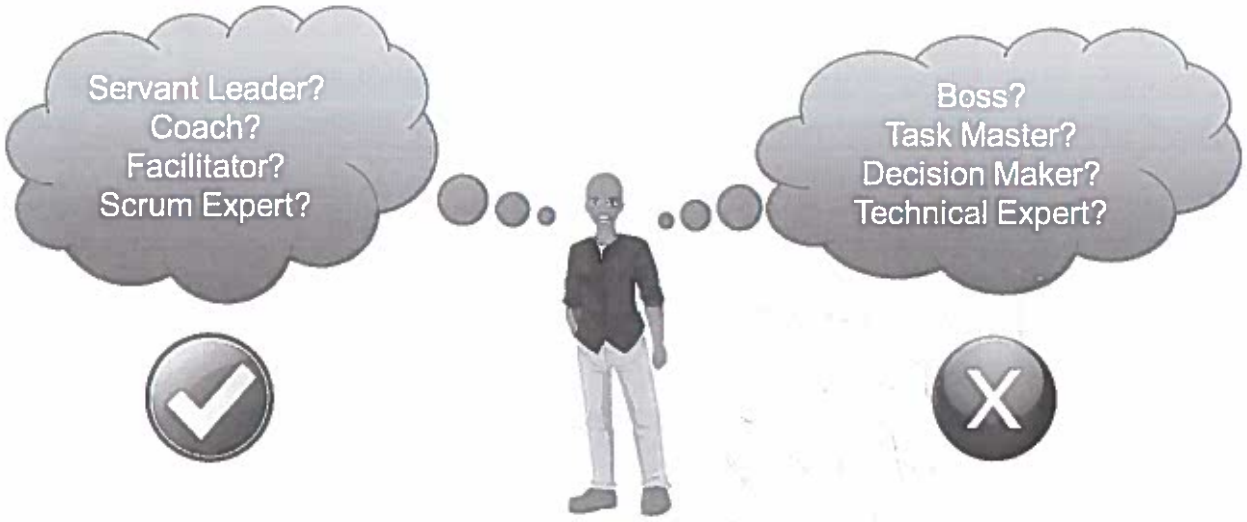
Discussion: Product Owner

Questions:

- Do you or will you have Product Owners for your organization's products?
- Are they empowered with the authority to make the final product decisions?

Tuckman
Forming
Stormy
Norming
performing

The Scrum Master



Why Not Agile Project Manager?

> "I wanted to highlight the extent to which the responsibilities of the Scrum Master are different from those of a traditional Project Manager."
>
> -Ken Schwaber
>
> *Agile Project Management with Scrum*

- In Ken Schwaber's book, *Agile Project Management with Scrum*, he addresses the question "Why not just call it Scrum Program Manager or Agile Project Manager?" The Scrum Master's responsibilities are different from those of a traditional Project Manager responsible for tracking tasks and time.
- Instead, the Scrum Master is a servant leader. The Scrum Master serves the Team; they are not anyone's boss.
- The Scrum Master coaches the Team; placing emphasis on people, coaching them to ensure they are working together as a Team, facilitating conversations so the Team comes up with the best outcomes.
- The Scrum Master is a Scrum expert—the master of the Scrum process—not expected to be a technical expert.
- The Scrum Master is not a decision maker—Scrum Masters do not assign work, decide what gets done when, or weigh in on Team's decisions, judge their estimates, change their estimates, etc.

Scrum Master & Scrum Team

Within the Scrum Team, the Scrum Master:

- Is dedicated to the Team full-time as a Scrum Master.
- Teaches and coaches Scrum.
- Does not work on/for the product.
- Facilitates as needed.
- Encourages disciplined engineering practices and approaches to work.
- Encourages collaboration between Product Owner and Development Team.
- Removes impediments and barriers for the Team.

- An easy area for new Scrum Masters to latch on to is the notion of "removing or escalating" impediments.
- This is similar to the issue tracking for resolution in traditional project management.
- The harder thing for new Scrum Masters to grasp, however, is the people aspect of the job—which is the most important part.
- The Scrum Master is the champion of the Scrum process, the master of the process, not the master over anyone.
- Their job is to foster collaboration with the Product Owner and the Scrum Team and to make the Scrum Team successful—not to interject their thoughts on estimate or design decisions, be a task master, etc.
- They need to shield the Development Team from any outside noise or interruptions so that the Team can remain focused on achieving Sprint goals that they committed to.
- The Scrum Master role is a full-time, dedicated role.
- This person cannot also be a tester, a developer, etc.
- One area inevitably suffers from lack of attention with their focus split and then the Development Team's goals aren't met—either way this is counterproductive and contradictory to the process.
- Does *not* perform multiple roles—is not a tester, developer, project manager, functional manager, etc.

Scrum Master & Organization

Outside the Scrum Team, the Scrum Master:

- Champions the Scrum process.
- Teaches Scrum to the organization.
- Coaches the organization to get the most benefits from Scrum.
- Works with other Scrum Masters to increase organizational agility.
- Helps spin up new Scrum Teams.

Outside of the Scrum Team, the Scrum Master:

- Acts as a change agent in the organization, championing the Scrum process and teaching those who are new to the framework or need education on Scrum (including Managers and Executives).
- Coaches the wider organization to get the most benefits from Scrum.
- Works with other Scrum Masters across the organization to increase organizational agility.
- Helps spin up new Scrum Teams.

Discussion: Scrum Master

Question: What challenges do you foresee a Scrum Master having in your Team? In your organization?

The Development Team

Development Teams:

- Are stable/long-lived and dedicated.
- Are self-organizing.
- Are cross-functional.
- Are accountable as a unit/Team.
- Are comprised of three to nine members.
- Participate in all Sprint Level Events.
- Demonstrate the product.
- Self-manage to execute the Daily Scrum and create/update Sprint Backlog & Sprint Burndown.

- Development Teams are ideally stable/long lived, and dedicated to one product or project at a time.
- There are no defined roles/titles in a cross-functional Development Team.
- They are self-organizing, which means that **no one**, not even the Scrum Master or Product Owner, tells the Development Team *how* to turn Product Backlog into Potentially Releasable Increments.
- The ideas of cross-functionality and self-organization will be hard for new Teams to grasp with years of being locked into traditional roles.
- They are cross-functional, which means they have all the skills needed (Dev, Test, Arch, DBA, UX, etc.), without depending on people outside the Development Team.
- Maybe not everyone can code, but are there people on the Team from testing, for example, who have some technical skills and can help automate test scripts?
- There are no sub-teams or silos in the Development Team, accountable as a unit/Team.
- Comprising three to nine members, who are accountable to each other.

- Participates in all Sprint Level Events (Sprint Planning, Daily Scrum, etc.), and often also participates in higher level planning activities too (varies by organization).
- Demonstrates the product (Potentially Releasable Increment) in Sprint Reviews.

- Self-manages to execute the Daily Scrum and creates/updates Sprint Backlog & Sprint Burndown daily and more often as needed.
- If the Development Team cannot create Potentially Releasable Increments, it creates a "mini waterfall" effect and builds up "technical debt":
    - This delays delivering value and puts the Product Owner in the unfortunate position at Sprint Planning of ordering work that "should" have been done previously, over the items that are next on the Backlog.

Discussion: Traditional Team Roles vs. Scrum Roles

Question:

- In a cross-functional, self-organizing Scrum Team, what happens to these traditional Team roles?
  - Business Analyst.
  - Technical or Systems Analyst.
  - User Experience.
  - Architect.
  - Developer.
  - Quality Assurance.
  - Technical Writer.
  - Project Manager.
- What challenges do you foresee in establishing the Scrum roles within your organization?

Assessment: The Players

Questions:

1. Who is responsible for creating and refining the Product Backlog?
    A. Development Team
    B. Product Owner
    C. Scrum Master
2. Who is responsible for teaching and coaching Scrum?
    A. Development Team
    B. Product Owner
    C. Scrum Master
3. Who is ideally stable and dedicated to one project at a time?
    A. Development Team
    B. Product Owner
    C. Scrum Master

Team Building – Collaboration

All Agile approaches:

- Have a common set of values, principles, and set of characteristics about Teams.
- Involve Team members working together, collaborating, and even sitting together, if possible.
- Involve Team members that plan, communicate, and discuss functionality:
  - During key meetings.
  - Outside of these meetings, as they are accountable to themselves and each other in the process.

- Traditionally, Team members have their own cubicles and/or offices and work on things that are assigned to them individually.
- With any Agile method, people are asked to work together in a collaborative way, co-locating if at all possible.
- Team members work together in the identified sessions but also are expected to continue designing, collaborating, developing, testing, etc., as needed outside of the sessions to complete working software with each Sprint or Iteration.
- Discuss with other class participants: What challenges do you see in transitioning from the way traditional Teams work to more of an Agile approach?

Team Building – Transition to Agile

When transitioning to an Agile way of working:

- The Scrum Master (Scrum) and the Coach (XP) are there to help the Team adopt the new process and to work together.
- An Agile Coach can be brought in to help move the Team in the right direction.
- Teams, Scrum Masters, and Coaches can leverage Retrospectives to improve the process.

- Some people make the transition to Agile very easily, and others are not able to change as readily.
- Some of the frameworks, like Scrum and XP, have a role identified to help the Team members focus on adopting the process and to work together better as a Team.
- Other approaches do not necessarily have a role dedicated to helping the Team in this way.
- If the Team is really struggling, the organization may look at bringing in an outside Agile Coach to help get them on the right track and improve their Agile transformation.
- There are other approaches that the Team and the Scrum Master or Coach can take to address these things on their own also.
- The Retrospective is one of the best tools to use for the Team to talk about what their process impediments are from Sprint to Sprint:
    1. A Retrospective can be held separately on the topic of the Agile adoption, or how the group is working together as a Team, to flesh out some things that are going well, that are not going so well, and what can be agreed to change as a group.
    2. There is no set time to hold that type of Retrospective—time will need to be made to do this if it is useful to the Team.
    3. Retrospectives will be covered in more detail toward the end of the course.

Team Building – Dysfunctions

Patrick Lencioni's book *The Five Dysfunctions of a Team*:

- Helps Teams overcome challenges and become more productive.
- Has a partnering Field Guide with several simple Team-building exercises.

- Team building is not necessarily an Agile topic. Since teamwork is so important in agility, however, it may be useful for Teams, Scrum Masters, managers, etc., to explore Team-building tools, books, articles, whitepapers, etc., that provide good ideas on the topic.
- A very popular book about transitioning a Team from one that may have challenges to become more of a productive, performing Team is Patrick Lencioni's *The Five Dysfunctions of a Team*. The book is easy to read and has a partnering Field Guide with several simple exercises that can be done in a session at the office or at offsite sessions. The exercises help Teams establish trust, identify common goals and values, learn to commit together as a Team, and learn to be accountable as a Team.
- For more information and ideas, see Patrick Lencioni's website: <http://www.tablegroup.com/books/dysfunctions>.

Let's Form Scrum Teams

Instructions:

1. Organize into Scrum Teams as directed by the Instructor.
2. If your business partner, Product Owner, and/or Scrum Master is in attendance, have them stay in these roles for the class simulation.
3. If you do not have a business partner, Product Owner, or Scrum Master in attendance, have someone on the Scrum Team play this role for the purpose of the simulation.
4. If your Team or organization will be adopting other methods or a hybrid approach, please experiment with the Scrum roles, try to apply them to your environment, and ask questions so we can ensure you get practical answers and guidance.

This page intentionally left blank.

Exercise: Self-Organizing Teams

Your instructor will provide direction for this exercise.

This page intentionally left blank.

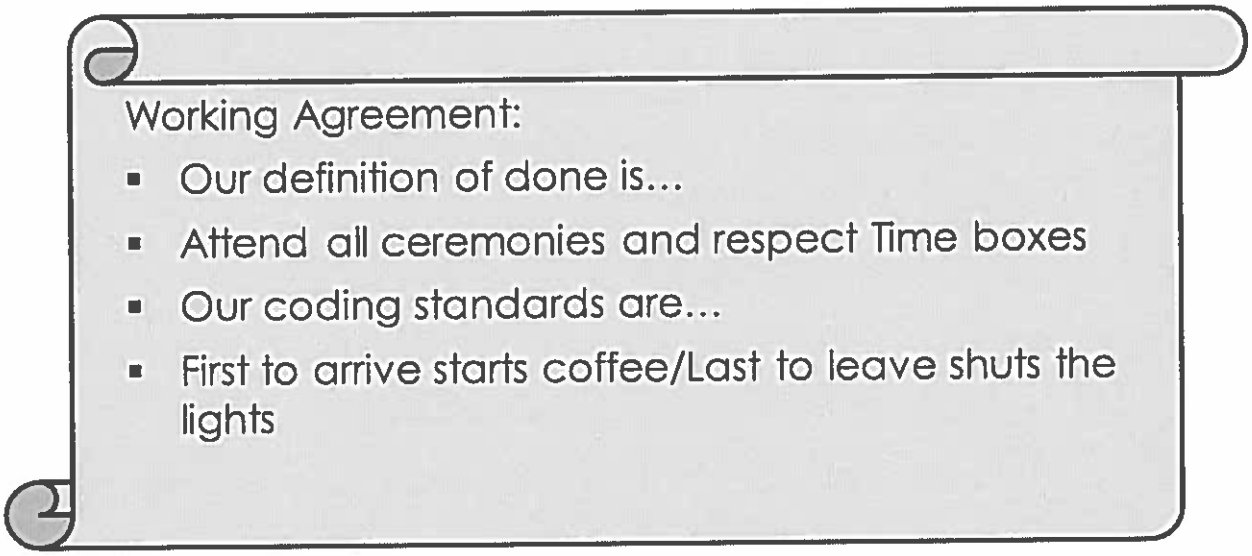
Working Agreements

A set of guidelines that the team agrees to follow.

Created by the Team for the Team.

Contains anything the Team feels is necessary to ensure healthy interaction.

Especially important for distributed Teams.



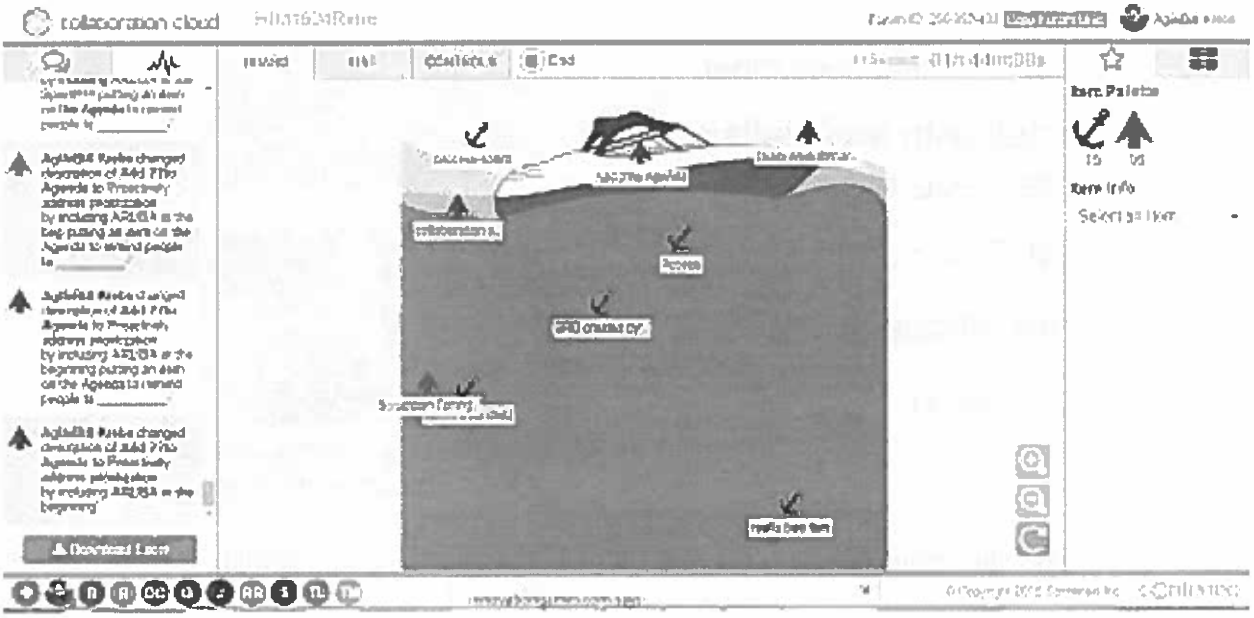
- Working agreements help to establish guidelines for heathy interaction among the team
- Anything is fair game for a working agreement as long as the team agrees that it will foster a positive environment.
- The last example given on the slide "first to arrive starts coffee / Last to leave shuts the lights" is included to illustrate this point. If break room etiquette or leaving lights on over night is an irritant to the team then it is worth calling out in the working agreement.
- More often items such as Definition of Done, Definition of ready, and specific processes or policies such as guidelines for test acceptance will be included.
- Working agreements are especially important when teams are distributed
    - Ensure that remote team members are included in ceremonies and decision making activities
    - Address communication challenges when teams are geographically dispersed
    - Provide visibility and transparency to and for remote team members

This page intentionally left blank.

Collaboration Techniques

How can you help the group produce more than just the individuals?

- Find tools that allow group input.
- Can be online, or face to face.



- There may be a wide range of tools and techniques for a given meeting. For example, retrospectives can be supported by Speed Boat, virtual notes as in the Retrium tool, and approaches like dot voting. Many of these have tools to support online, as well as face to face needs.
- Speed boat, shown on the slide, is one example of Collaboration Frameworks. Other formats like *Prune the Product Tree*, or *Buy a Feature*, or *20/20* let you prioritize work. Some, like *Product Box*, let you form high level needs for your products. Some platforms also let you design your own formats.

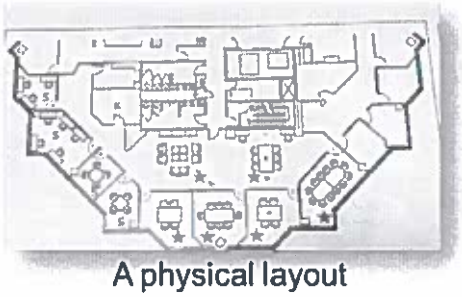
Resources:

- Tools for distributed retrospectives, prioritization, and analysis: <http://conteneo.co/collaboration-frameworks/>
- Tool for distributed retrospectives: <https://www.retrium.com/>
- Book – *Gamestorming* by Dave Gray and Sunni Brown
- Book – *Innovation Games* by Luke Hohmann
- Book – *Agile Retrospectives* by Derby and Larsen
- Book – *Collaboration Explained* by Jean Tabaka

Discussion: Work Environments and Collaboration

Which environments and communication styles work best?

- Common Room?
  - Information Radiators.
  - People Facing each other.
- Cubicles with low walls; people close together?
- Virtual Co-Location tools?
- Private offices or cubicles?
- Phone calls?



A physical layout

A virtual layout

- High Bandwidth communication is our goal in Agile. Given geographic constraints many companies have, how can we eliminate as many barriers as we can to good dialog?
- **Physical workspace designs.**
- Communication is better when we have:
  - Common Team room (as opposed to individual cubes).
  - Large information radiators (such as burn down charts, task boards, and build progress indicators).
  - Can people be in the same room, same aisle, same floor, or same building? Some Teams find if people have to walk too far, they end up messaging instead.
- **Virtual workspaces**
  - Do virtual Teams need to think about their workspace configuration too? Some teams use electronic tools to categorize topics and messages (such as chat topics in Slack, or Room layouts in Sococo). What working agreements does your team have about how to communicate when?
  - There is a branch of tools that will do better than just screen sharing. Some examples include:
    - Skype: allows you to communicate with distributed Team members and share your screen.
    - Slack: allows you to work efficiently with distributed Team members, categorize topics and messages, and share documents.
    - Sococo: a virtual co-location tool that allows you to set up a virtual office.

This page intentionally left blank.

Assessment: Team Building and Collaboration

Questions:

1. True or False? It is not important for Agile teams to collaborate.
2. True or False? Dysfunctional teams lack commitment and avoid accountability.

Agile Planning: The Vision

Levels of Agile Planning

Vision: An opportunity that we have with clients or in the market—a product strategy.

Roadmap: Lays out a high-level plan for when we want to realize this Vision.

Release Planning: Tactical meeting to discuss realistic dates that we can Release the product.

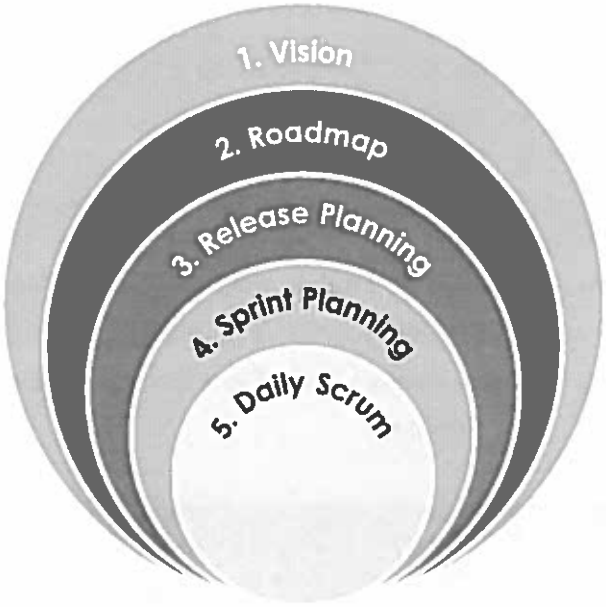
Sprint Planning: The Scrum Team plans to incrementally meet product requirements.

Daily Scrum: Enables the Dev Team to inspect and adapt.

- It is a common myth that Agile methods involve a lack of planning, with Teams diving into Sprints and developing without any upfront planning.
- Regardless of the Agile method, a key input into what happens at the Team level is planning at higher levels in the organization.
- For the purposes of this course and the subsequent simulation exercises, we will be using the more industry-adopted five Levels of Agile Planning.
- The more widely adopted five levels of planning note that Vision and Strategy are the same step; Portfolio Planning is addressed by Roadmap and Release Planning; and Continuous Planning is redundant as a separate step, since planning occurs daily to inspect and adapt.
- Depending on the company or author, Agile Planning has been described as having either five or six levels.

Agile Planning – Product Vision

- Is the overall goal or purpose for the product or service.
- Is a high-level sketch of what the product or service looks like.
- Communicates the essence of the product in a concise manner.
- Reflects common agreement and understanding among stakeholders.



- The Product Vision is the top layer of Agile Planning.
- The Vision paints a picture of the future that draws people in.
- Capturing the Vision is facilitated by the Product Owner or Customer working with any stakeholders, leaders, Subject Matter Experts (SMEs), etc.
- Most importantly, the Customer or Product Owner shares the Vision with the Development Team who will be doing the work to realize that Vision.
- As part of the Product Vision, consider the following:
    - Target: Who are the users or customers?
    - Needs: What problems will this solve? What benefit does it provide?
    - Product: What is the product? What differentiates it from other products?
    - Value: What are the business goals? How does this product benefit the company?

Share the Vision

To share the Vision, the Product Owner will:

- Discuss and Communicate the Vision to:
  - The Development Team.
  - The Stakeholders.
  - The Organization.
- Choose a method to document the Vision:
  - Project Charter or similar document.
  - Business Case, Cost/Benefit Analysis, or Business Model Canvas.
  - Template.

Your Themes, Epics, and User Stories should align with the Vision.

- There is no right or wrong way to develop the product Vision. The important thing to note is that one exists and everyone understands and agrees on what it is.
- The Product Owner is responsible for articulating the Vision. The PO works with key stakeholders, Subject Matter Experts (SMEs), and the Development Team.
- The Development Team should know what the Vision is and where it is. If they don't then the Product Owner has not clearly articulated or shared it. Some Teams have Vision statements printed out and posted visibly on the wall of their Team area, next to task boards.

Elevator Pitch Template

A widely adopted template to document the Product Vision is Geoffrey Moore's Elevator Pitch from his book "Crossing the Chasm"

FOR <target customer>
WHO <state of the need>
THE <product name>
IS A <product category>
THAT <key benefit>
UNLIKE <primary competitor>
OUR PRODUCT <further differentiation>

From Geoffrey Moore, *Crossing the Chasm*

- Although there is no standard template, or one way to capture the Product Vision, many in the Agile community have adopted Geoffrey Moore's "elevator pitch" template from his book *Crossing the Chasm*.
- It is brief, to the point and easy for most Product Owners to be able to use to succinctly capture the Vision and share it with the Development Team.
- Its name comes from the fact that someone should be able to communicate this in a few minutes, or the time it takes to ride in an elevator to answer the question "what are you working on?"

Exercise: Vision

Instructions:

1. In your Teams, create a Vision for your product or service using the Geoffrey Moore template.
2. If your Product Owner is in attendance, he or she should drive this discussion with input from the rest of the Team.
3. Capture the Vision on paper and be prepared to share what you came up with.

FOR <target customer>
WHO <state of the need>
THE <product name>
IS A <product category>
THAT <key benefit>.
UNLIKE <primary competitor>
OUR PRODUCT <further differentiation>

From Geoffrey Moore, *Crossing the Chasm*

Discussion: Vision

Questions:

- In your organization, has someone been responsible for determining the Product or Project Vision? If yes, who is this? If no, what challenges do you see there?
- In order for your Scrum Team or organization to successfully adopt Agile, how important is it to understand the Vision—what you're actually being asked to deliver?

Assessment: Agile Planning: The Vision

Questions:

1. True or False? The Development Team discusses and communicates the Vision to the stakeholders and organization.
2. True or False? Geoffrey Moore's Elevator Pitch is a widely adopted template to document the Product Vision.

TEKsystems
Our people make IT possible.

Agile Planning: The Roadmap

Section 9

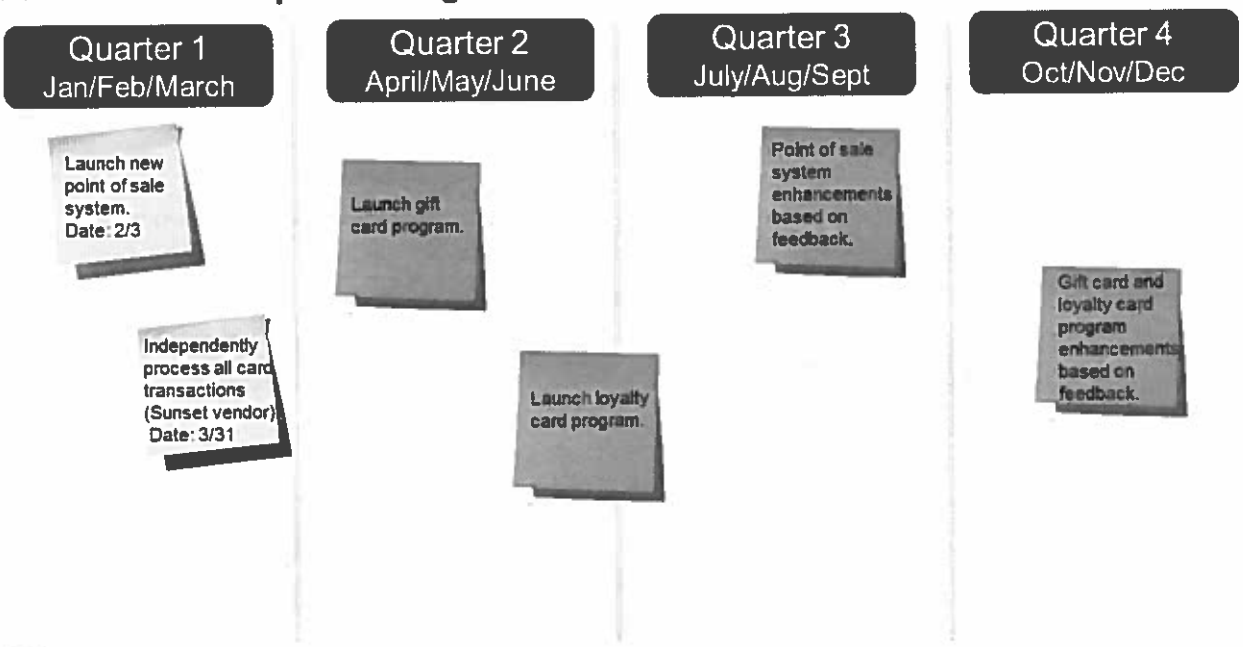
Agile Planning – Product Roadmap

Roadmap Planning:

- Typically takes place after establishing the Vision.
- Ensures priorities are aligned at all levels including the Scrum Teams executing the Vision.
- Is not necessarily a step in Scrum or other Agile methods, but has been widely adopted.
- Is visualized in a Product Roadmap.

- Vision Planning can occur at any time in an organization's fiscal year. Maybe this was an idea previously identified or it's in response to a competing product or service or the result of research, customer focus groups, etc. The Vision or idea then drives the Roadmap.
- At a high level, the organization must decide what its priorities are for projects, initiatives, etc. It is important to note that at the Roadmap level, we have not engaged the people who will be doing the work who can best tell us how long the work will take. For that reason, Roadmap plans are high-level goals and not exact dates. Instead, it is expressed at the quarter level (Q1, Q2, etc.) or perhaps the time of year (e.g., first half of the year, second half of the year).
- The Product Owner is responsible for keeping the Roadmap current and visible to the Team executing it.
- The Product Roadmap provides the link between the Product Vision and the strategy for executing that Vision and it must be communicated to the organization, stakeholders, and the Development Team.
- Keep the Roadmap simple and easy to understand.

Product Roadmap: Evolving Across Releases



Forecasted plan, dates, and work

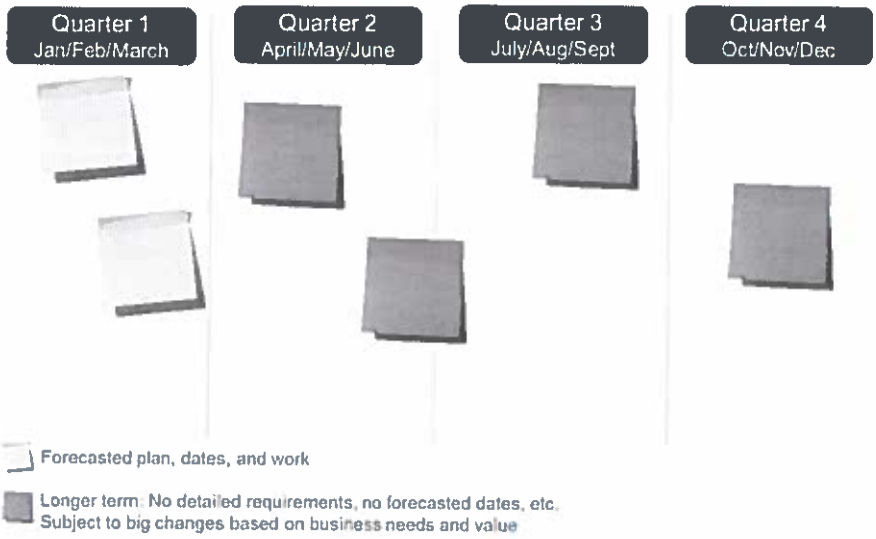
Longer term: No detailed requirements, no forecasted dates, etc.
Subject to big changes based on business needs and value

- The Product Roadmap can help an organization or Scrum Team:
  - Show how the product will evolve across Releases or versions.
  - Facilitate dialog between stakeholders, Product Owner, and the Scrum Team.
  - Simply state essential and ordered high-level milestones for the product(s).
  - Focus on the Product Backlog from Vision through execution using the high-level timeframe outlined on the Roadmap.
- The Roadmap should continually be evaluated, shifting as new things come on to the plate, others drop off, etc.
- Product Roadmap Example: On this slide is an example of a Roadmap where a Team or Teams has responsibility to release one, versioned product:
  - The quarter that is yellow is the one closest to us (nearer timeframe), so those initiatives already have budget and are either ready to hold Release Planning or have held Release Planning.
  - The quarters that are in blue are further out and may change based on feedback we get from customers, market reaction, competitors reaction, etc.
  - As those further timelines draw closer, the Product Owner makes decisions on what features to move forward with in the "next version," holds Release Planning, and secures funding to kick off that next "project phase."

Exercise: Roadmap

Instructions:

- In your Teams, create a high-level Roadmap for your product:
  - Please note that the Vision provides key input into this level of planning.
  - If the Product Owner is in attendance, have them drive this discussion, taking into account the organization's fiscal year or Release cycle.
- Capture the Roadmap on paper and be prepared to share your results.



Assessment: Agile Planning: The Roadmap

Questions:

1. True or False? The Product Roadmap can help an organization show how the product will evolve across Releases.
2. True or False? The Product Roadmap is finalized before the project starts and does not change.

TEKsystems
Our people make IT possible.

Agile and the Customer

Agile Processes and the Customer

The customer is at the heart of Agile:

- Our highest priority is to satisfy the customer with early and continuous delivery of valuable products or software.
- Agile processes use change as a competitive advantage for the customer.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business owners and developers must work together daily throughout the cycle.

- There can be a misperception about Agile being "for developers" or that it is an "IT process."
- It is actually a business or customer-driven process.
- Without the Product Owner or a customer driving the Vision and the business priority ensuring that we will deliver value with each Iteration, Teams are left to try to guess about what to focus on.
- Agile welcomes changing requirements, even late in development.

Who is the Customer?

The customer is the central focus of Agile and is actively involved throughout the cycle.

- In order to best define the Product Vision, we need to focus on customers:
  - Who are they?
  - What's important to them?
  - How will they use the product or service?
  - What are their priorities?
  - Are there any characteristics we need to consider, such as technical or computer proficiency, or lack thereof?

- The customer is the central focus of Agile and is actively involved throughout.
- A useful exercise that is often overlooked in Agile Planning is spending some time talking about who the end user or the customer for the product or service will be.
- If we understand more about the user, it can flush out important details about the feature or functionality.

User Roles

- With Agile methods, requirements are conveyed by the customer and represent the user's point of view.
- Unique user perspectives help flesh out emerging requirements and help define what we mean by "done."
- Identifying users can be as simple as categorizing the user roles that exist for a product or service:
  - For example, for a point of sale system at a retail store, the user roles may be "cashier," shift leader," and "store manager."

- Identifying users can be as simple as categorizing the user roles that exist for the product or service.
- For example, for a point of sale system at a retail store, some user roles may be cashier, shift leader, and store manager.

What is a Persona?

- An imaginary character who represents a user role.
- Useful when you do not have easy access to real users.
- Creating a persona involves more than coming up with a name for the character:
  - What job do they do for a living?
  - What demographic do they represent?
  - What is their educational background?
- Major characteristics of a *user* or *group of users*.
- A persona can be a *positive* or *negative* character to demonstrate desired or undesired functionality.

- Identifying user roles is a great start. For more important user roles, there's a set of named characteristics that takes this a step further, called personas.
- The persona technique was first introduced by Alan Cooper, to define an archetypical user of a system; an example of the kind of person who would interact with it.
- Personas are used to represent different user types within a targeted demographic identifying the goals and behavior of this hypothesized group of users.
- A persona is a fictional representation of a set of user characteristics; it is not any actual user's name.
- Creating personas is not part of Scrum or Agile, but many find it to be a useful technique in understanding requirements from the user's point of view—which is an idea at the core of any Agile approach.

Example Persona

Connie the Cashier:

- Connie is a cashier in a retail department store.
- She has worked at the store for 5 years.
- Connie is a part-time, evening, and weekend employee.
- She has some computer experience, but mostly for home use in word processing, checking email, and shopping online.
- Connie has a journalism degree and has been submitting writing samples to magazines to try to land a full-time role as a journalist.

- "Connie the Cashier" is an example of a persona.
- The idea is that whenever someone on the Team refers to "Connie the Cashier," these characteristics come to mind and there's an understanding of the level of detail needed in the requirements, or the persona can be used to focus the requirements discussion.
- Discuss with other class participants: Do you find this technique useful? Is it one that you have tried in your Teams?

Example of Negative Persona

Harry the Hacker:

- Harry makes his money by:
  - Setting up online virus scams.
  - Hacking into unsecured databases to obtain Social Security or credit card numbers.
- He has a high degree of computer proficiency and has advanced knowledge of security systems for online applications.
- He works a day job that he would like to quit.

- Personas can also be negative to illustrate a point or to ensure that the Product Owner understands the implications of not prioritizing something that may—at a glance—be considered technical, like the need for security.
- "Harry the Hacker" is an example of a negative persona.
- The idea is that whenever someone on the Team refers to "Harry the Hacker," these characteristics come to mind and there's an understanding of the level of detail needed in the requirements, or the persona can focus the requirements discussion.
- Giving the character a name and talking about his or her less than noble intentions can give that requirement new light, and it can get prioritized more appropriately along with other requirements.

Exercise: User Roles & Personas

Instructions:

1. In your Teams, identify the product, service, or project you will use for our simulation.
2. Discuss the user roles and personas for this product or service and document any that you come up with.
3. If your Product Owner is in attendance, he or she should drive this discussion with the Team and SMEs providing input.

Assessment: Agile and the Customer

Questions:

1. An imaginary character who represents a user role is called a:
   A. User
   B. Persona
   C. Customer
2. True or False? With Agile methods, requirements are conveyed by the customer and represent the user's point of view.

User Stories and the Product Backlog

User Stories

- A User Story is a requirement or feature expressed *briefly* from the user's perspective.
- Stories are a promise to continue the conversation.
- User Stories describe:
  - Who has a particular need or want.
  - What that need or want is.
  - Why they need it.

- A *User Story* is a requirement or feature expressed briefly from the user's perspective. The few lines of the User Story are not meant to be the requirement in its entirety.
- It is meant to be the start of an evolving conversation.
- User Stories describe who has a particular need or want, what that need or want is, and why they need it:
  - The Why is important! Without it, we have not captured the business value or need and have taken away one of the items that helps show priority.
- In traditional processes, like waterfall, a large document took the place of the conversation.
- Because we value individuals and interactions over processes and tools and working software over comprehensive documentation, we want to collaborate and talk with each other about the Story, documenting the results of the conversation.
- User Stories are not an official part of the Scrum framework, but they have become common in the Agile community.

What is the Product Backlog?

Order

Story #	Story Points	Description
2	3	Stand up dev server for project
1	8	User Story 1
5	5	Fix support ticket #3001
3	3	Research display options
4	5	User Story 2
6	2	Upgrade to Flex version 4
9	8	User Story 3
8	1	User Story 4
7	13	User Story 5
10	?	Fix defect RA2441

- The Product Backlog is the high-level to-do list for the product, and is the responsibility of the Product Owner. It is the single source of requirements for changes to the product. The Product Backlog:
  - Is an ordered list of all the work we have to do to release the product.
  - Includes the features, functions, User Stories, requirements, enhancements, and fixes that constitute changes to be made to the product in future Releases.
  - Can also include foundational work, constraints, or analysis we need to perform.
- Items on the Product Backlog are frequently, but not always expressed as User Stories. Non-functional requirements, support tickets, technical considerations and other items that can't be expressed from the end user's point of view may simply be referenced in plain language.
- Each item in the Product Backlog has a rough estimate of the business value and development effort:
  - The Product Owner orders each item on the Backlog in relation to the other items on the Backlog.
  - Items in the Product Backlog are ordered based on considerations such as risk, business value, dependencies, and date needed. Items higher in the list have more detail, while items further down the priority list are still broadly and imprecisely defined.
- The Product Backlog does not address how the work is completed, only the scope of work that needs to be created.

This page intentionally left blank.

The 3 C's of User Stories

Card	Traditionally captured on a note card; cards may be annotated with estimates, notes etc.	As a user, I want to login and gain access to the www.esCARgot.com website, so that I can purchase a car.
Conversation	Details captured in conversations with Product Owner.	What if my account is expired? / Will I remember my login? / Can I reset my password?
Confirmation	Acceptance Criteria confirms that the Story was coded correctly.	Acceptance Criteria: 1. Expired accounts fail. 2. It remembers the login, but not the password. 3. I can reset my password.

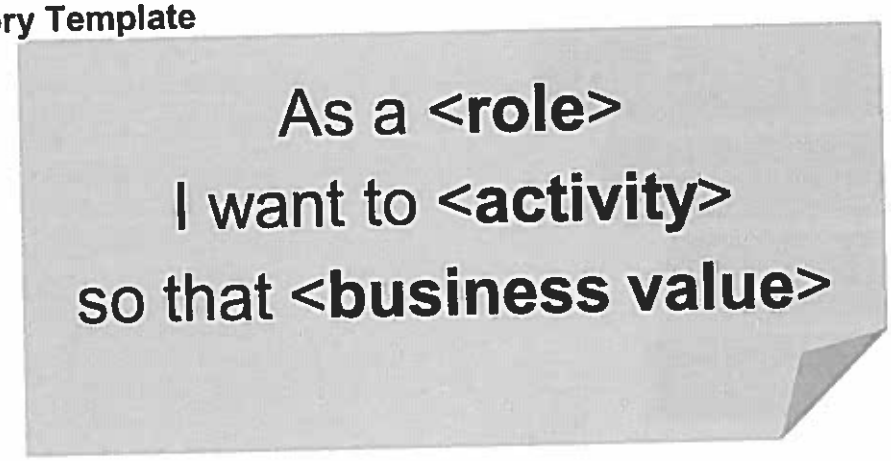
Source: XP Magazine 8/30/2001, Ron Jeffries

- A User Story is a short description of something your customer will do when they come to your website or use your product or service. It is focused on the value or result they get from doing this thing. In other words, it is a tool and technique used in Agile software development to express requirements. It captures a description of a software feature from the end-user perspective and is short enough to be completed in one Sprint.
- According to Ron Jeffries, there are Three C's of User Stories: Card, Conversation, and Confirmation:
    - Card: The card is a short statement of functionality usually captured on an index card or sticky note for visual organization, and may be managed in software, such as Rally. It describes the type of user, what they want, and why. The statement should use the language of the customer so that it is clear to both the business and the Development Team what the customer wants and why. Physical cards are excellent when eliciting, prioritizing, and organizing User Stories, but the card is a reminder or placeholder for a deeper conversation. A card is not meant to be just the Agile version of a long, detailed written requirement.
    - Conversation: The few lines of a User Story are not meant to be the requirement in its entirety. They are meant to be the start of an evolving and ongoing conversation. A User Story requires collaboration and conversations between business owners and the Development Team to clarify the details as the code is developed. Because we value individuals and interactions over processes and tools and working software over comprehensive documentation, we want to collaborate and talk with each other about the Story, documenting the RESULTS of the conversation, the Acceptance Criteria.

- Confirmation: User Stories are given more detail as the Conversation is held and things evolve and emerge. These details, called Acceptance Criteria also could be understood as high-level test cases—the cases that must be met for the business to accept a given Story.

- The Development Team determines how to satisfy the requirements of the User Story.

A User Story Template



- **Role** ("Who"): Represents who is performing the action. It should be a single person, not a department. It may be a system if that is what is initiating the activity.
- **Activity** ("What"): Represents the action to be performed by the system.
- **Business Value** ("Why"): Represents the value to the business. "Why is this Story important?"

- The three sections of a User Story are the Role (Who?), the Activity (What?), and the Business Value (Why?).
- The Why is important. Without it, we have not captured the business value or need and have taken away one of the items that helps indicate priority.

INVEST in Good Stories

I Independent

N Negotiable

V Valuable

E Estimable

S Small [Small enough to fit in a Sprint]

T Testable

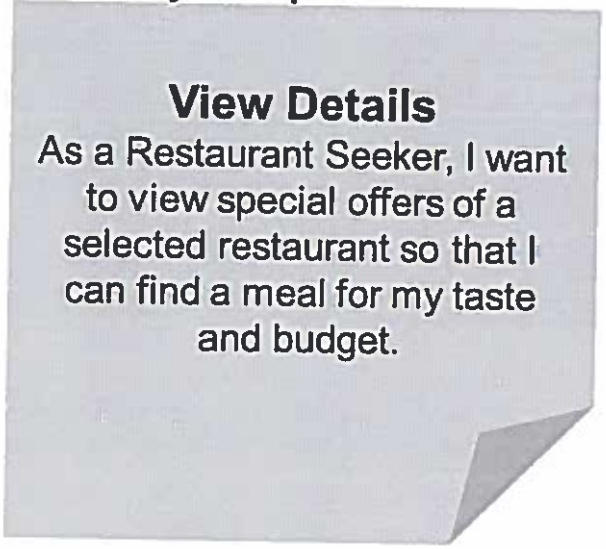
Source: Bill Wake, http://xp123.com/articles/invest-in-good-Stories-and-smart-tasks/

There are no hard and fast rules for writing User Stories, so how do you tell whether or not a User Story is good? The INVEST acronym is simple way to evaluate User Stories. INVEST is an acronym for the characteristics of a good User Story:

- **Independent.** Ideally the User Story should not be dependent on other Stories. We try to keep the functionality discrete so that we do not set up intricate dependencies that prevent us from getting to "done" by the end of the Sprint. Ask: Is the Story dependent on other Stories?
- **Negotiable.** User Stories are not cast in stone. They are reminders to have a conversation and will evolve as the Product Owner and the Development Team discuss them. User Stories should have enough information to capture the essence of the feature without requiring too much collaboration for the basics, but they should not have too much detail because then the conversations may not happen. Ask: Is there enough information? What other information would I need?
- **Valuable.** User Stories should have value to the user or owner of the solution. If there is no "why" statement that expresses the business value, we may not need to work on this feature. Why work on something that will realize no value? Ask: Is the Story valuable to the user or the owner? What is the value?
- **Estimable.** The Development Team should be able to estimate the User Story. A User Story may not be estimable if it is too vague, doesn't have enough information, or is too big. Ask: Could you estimate this Story? If not, is the Story too big? Too vague? Missing too much information?
- **Small.** User Stories should be small, but not too small. Sometimes this is referred to as right-sized. By the time you include them in a Sprint, User Stories should be small enough that several can be completed in one Sprint, but not so small that they could be considered tasks. Ask: Can the Story be completed in one Sprint? Is the Story too small?

- **Testable.** User Stories should be testable. We must have a way of checking the Story against the Acceptance Criteria so that we know we have achieved our Definition of Done. Ask: Is the Story testable?

User Story Examples



Here are some additional User Story examples of the INVEST model at work:

- Even though these Stories are part of a larger application or website, they can be completed independently.
- If someone asks the Product Owner, "What do you mean by ratings – food? Ambiance? Service?" the Product Owner may realize that this is a vague Story so he or she clarifies in the language of the Story or in the Acceptance Criteria as a result of the Negotiation with the Development Team.
- Both Stories are Valuable, as we're told why they are important to the end user or business.
- Both Stories are able to be Estimated, so they are not Epics but are well-sized User Stories.
- Both Stories are Small or Sized right, as they should be able to be completed within a Sprint.

Exercise: Create User Stories

Overview: Create 3-5 User Stories for the product you chose earlier today. Use your Product Vision and the roles and personas you identified to help.

Instructions:

- Break into your groups.
- Based on your Product Vision and the roles and personas you identified for your product, create 3-5 User Stories. (More is better).
- Be prepared to share with the class.

As a **<role>**

I want to **<activity>**

so that **<business value>**

User Story Conversations

User Story Details? Ask the Product Owner.

- Are there other options for landing at this particular page?
- Are we limited by user security or access?
- Do we need to account for error handling or the "sad path" in addition to the "happy path"?
- How do we confirm the functionality works correctly?
- Do you think we have all of the Acceptance Criteria for this Story?

- The initial User Story should fit on an index card or sticky note capturing the Who, What, and Why in a brief description that everyone understands, but it likely doesn't have all the details you need to actually work on the Story.
- That's because Stories are not contracts or set in stone—they are emergent. Details are captured as they unfold. And, they unfold as the Development Team has conversations with the Product Owner.
- The questions that appear on this slide are typical ones the Development Team would ask the Product Owner to clarify the Story.
- Development Team members should capture the answers as Acceptance Criteria. Depending on how the User Story is documented, you might put the Acceptance Criteria on the back of the User Story sticky note or card, in a Word doc, or in a tool where Story information is kept (e.g., Rally).
- The Story should be visible and accessible to the Development Team when they start to work on it.

Acceptance Criteria

Acceptance Criteria:

Given that I searched for a list of restaurants, I expect that:

- Restaurant Name, Location, and Phone Number will be displayed.
- I will be able to sort the list by Location.
- I will be able to sort the list by Rating.

- Instead of replacing the conversation with an upfront, detailed document, we allow the details to emerge through conversations.
- Acceptance Criteria are the result of the conversations that we had about the User Story.
- Acceptance Criteria help the Scrum Team know when the Story is complete.
- Acceptance Criteria aid the Development Team with testing, specifically test automation.

- In traditional processes, we received large documents to read through at the beginning of a project.
- Often times there were other roles to go through as "go betweens" to get the answer from the customer, end user, or voice of the customer.
- Agile strips the back and forth out and has the voice of the customer, the Product Owner, in the driver's seat and accessible to the Development Team.
- Agile also forces the conversation by not having a large document available at the beginning of the project.
- We document the results of the conversation—we do not replace conversation with documentation.
- As conversation takes place, emergent details should be captured.
- These may be valuable Acceptance Criteria that we want to ensure the Development Team who is doing the work understands.
- There is no rule, formula, or prescribed template for how to do this.
- Scrum Teams are empowered to do this in whatever manner works best for them whether a document is leverage, an Agile project management tool, notes written on the back of the initial Story index card, etc.
- The important thing is to have Acceptance Criteria so the Development Team knows.

- They need to define what the Definition of Done is by the end of the Sprint.

Exercise: Create Acceptance Criteria

Overview: For the User Stories you created in the last exercise, determine your Acceptance Criteria.

Instructions:

- Break into your Teams.
- For each User Story, create the Acceptance Criteria.
- Ask yourself: How do you know you're done? How do you know it works? And based on the answers, what are a few things you would determine as Acceptance Criteria.
- Complete criteria for as many Stories as you can.
- Be prepared to share.

Assessment: User Stories and Product Backlog

Questions:

1. What helps the Scrum Team know when the Story is complete?
    A. The Business Value or "Why" of the User Story
    B. Acceptance Criteria
    C. INVEST
2. What are the three Cs of a User Story?
    A. Card
    B. Confirmation
    C. Collaboration
    D. Conversation
3. Which of these can be items on the Product Backlog?
    A. User Stories
    B. Bugs and Defects
    C. Non-functional Requirements
    D. All of these

Definition of Done

"DoD"

Story Acceptance Criteria vs. Definition of Done

Acceptance Criteria:

- Describes correct behavior of functionality—proving it works.
- Applies to only one Story.
- Product Owner has the final say.
- Created during refinement sessions.

Definition of Done:

- Describes a minimum software quality standard.
- Applies to all Stories.
- Decided/created by the Scrum Team.
- Created prior to Sprint 1.
- Should eventually represent a "Releasable Increment" standard.
- Continuously improved over time and during Sprint Retrospective.

- As you know, Acceptance Criteria describes the correct behavior of functionality, proving it works. Acceptance Criteria emerge during conversations with the Product Owner and Stakeholders and applies to only one Story. Because the Product Owner owns the User Story, the Product Owner has the final say on the Acceptance Criteria.
- That's great for the User Stories, but how do we know we're really done? That's where the Definition of Done (DoD) comes in. DoD is created prior to the first Sprint. It describes the minimum software quality standard and should eventually represent a Releasable Increment standard. DoD applies to all Stories and is continuously improved over time and during the Sprint Retrospective.

What is Definition of Done (DoD)?

A Scrum Team must decide on and document its DoD:

- It is important that everyone have the same understanding of DoD:
  - "Done" is not enough—done is different to different individuals.
  - DoD is what makes a Story ready for Sprint Review.

"DoD" may vary by role

Role	Example of "done"
Coder	**DoD** may mean code complete.
Stylist	**DoD** may mean everything looks good and the style sheets have been followed.
Quality Assurance	**DoD** may mean that all tests have been executed.

- The Scrum Team (Development Team and Product Owner) decide on DoD.
- The Scrum Master facilitates the process but does not vote on DoD.
- It is the Scrum Master's duty to validate with the Development Team upon Story completion that the Story has met DoD before presenting it to the Product Owner.
- The Product Owner will use the DoD to determine if a completed Story is accepted or rejected.
- The Product Owner has the authority to accept or reject a completed Story.

What is Definition of Done (DoD)? (Cont.)

A Scrum Team must decide on and document its DoD:

- The Scrum Team defines DoD.
- The Development Team uses the DoD to hand over the Story to the Product Owner for acceptance.
- Once consensus has been reached, the Scrum Master records the DoD and posts it to be seen by anyone within the organization (full transparency).

Example:

This is an example of a list of items that would be checked off after determination is made that they are done:

- ✓ Coded
- ✓ Checked in
- ✓ Styled
- ✓ QA'd
- ✓ Product Owner over the shoulder checked
- ✓ Built
- ✓ Promoted

- The Product Owner must participate in the decision of the DoD and gain consensus with the Development Team on what will be accepted and what will be rejected.
- The DoD will vary from Development Team to Development Team.
- The final DoD for a Team should be posted and used to determine acceptance of a completed Story.
- If multiple Teams work on the same product, then all of those Teams should mutually create the DoD.

Suggested Steps to Create Definition of Done (DoD)

- This is one way to create a Definition of Done. The important thing is to think through what done means and for the Team to come to agreement.
- Suggested steps to create the Definition of Done:
    1. **Brainstorm:** Write down everything essential for delivering on a feature, Iteration/Sprint, and Release; for example, code is complete and checked in, Product Owner signoff, and updated release notes might all be essential pieces of delivering on a feature. Write one item per sticky note. As you brainstorm, think about what it means for that feature to be shippable to ensure that you catch everything that is essential for delivery. Think also about each of the different roles—done for a tester might bring up different items than done for a developer.
    2. **Identify items that can't be checked every Iteration or Sprint:** Look at each of the items on the sticky notes and identify whether or not they can be done at every Iteration/Sprint for each feature being delivered. If they can't be, remove them from your Definition of Done.
    3. **Capture impediments:** For each item that cannot be checked every Iteration or Sprint, discuss the obstacles that keep the Team from delivering this each Iteration or Sprint deliverable. Frequently, these obstacles create issues such as having an unpredictable release period after the last feature is added. Even if the obstacles can't be removed right away, over time, they can be removed and the item can be included in the Team's Definition of Done.
    4. **Commitment:** Get a consensus on the Definition of Done. Go through each item that is completed each Iteration or Sprint.

Class Exercise: D

Overview: In your Team

Instructions:

- Have one persor rest as developer
- Be prepared to s

Definition of Done
- Develop Unit Test
- Coded, documented
- Peer Reviewed by other Developer
- Peer Reviewed by Security Analyst
- Functional System Testing
- UAT
  - affiliate API testing
  - developer API testing
- UI/ux Review
- Product owner review

Test Geo - redundancy
Pass Unit testing
Compiled / Built
Deployed / Promoted
Post deployment review

Assessment: Definition of Done

Questions:

1. True or False? The Definition of Done is created by the Product Owner.
2. True or False? Definition of Done is what makes a Story ready for Sprint Review.

Manage the Product Backlog

Review: Product Backlog

- The Product Backlog: single source of requirements for changes to the product. The Product Backlog constantly evolves, adding details that stem from ongoing conversation.
- To prepare for Release Planning, we need a basic Product Backlog. It should include at least a minimal amount of detail in a list of Product Backlog Items ordered by business value.
- Product Backlog management includes communicating items, ordering, optimizing for value, ensuring transparency of the Product Backlog, and ensuring that the Dev Team understands the items to the level needed:
  - The Product Owner may do the above work, or have the Dev Team do it; however, the Product Owner remains accountable and responsible for the completion of the work.

- The Product Backlog is the single source of requirements for changes to the product:
  - Backlog Items are an ordered list of the features, functions, User Stories, requirements, enhancements, and fixes that ***might*** be in the product someday.
- The Product Backlog can live in a spreadsheet, Word document, Agile project management tool, etc.
- There is no prescribed template or formula for the Product Backlog.
- Product Owners are empowered to capture this in whatever medium works best for them and the Development Team.
- As the Product Owner, Development Team, and key stakeholders discuss the Product Backlog Items, this collaboration may change the order of the Items as we uncover dependencies, priority, business value, etc.
- Some may use a document, a spreadsheet, an Agile project management tool, a list on a shared collaboration tool, etc.

Exercise: Build the Initial Product Backlog

Instructions:

1. In your Scrum Teams, create a Product Backlog.
2. Product Backlog Items can include User Stories, non-functional items, fixes, technical foundation items, etc.
3. Use sticky notes, index cards, or paper to capture your Product Backlog Items.
4. Create between 10 and 20 Product Backlog Items, splitting them as necessary.
5. Do not worry about ordering or Estimation just yet—we will learn those techniques in subsequent exercises.

Ordering the Product Backlog

Items are small, higher priority, have lots of details, and will be worked on soon.

Items are large, lower priority, and have few details. They will be better defined and right-sized as they move up the Backlog.

- Jim Johnson, Chairman of Standish Group, reported at the Third International Conference on Extreme Programming (XP2002) that in typical software systems 64% of features are never or rarely used in reality. The most effective way to reduce software cost is to prioritize and deliver only highly used value features.
- Prior to Release Planning, the Product Owner orders the Product Backlog by Business Value. Business Value can be determined by a financial model or a combination of a financial decision and what will bring customers the most satisfaction or value, or drive the most use. Highest priority is given to Stories with the highest business value.
- The Product Owner can use whatever technique for arriving at Business Value that works best for them and the organization.
- The Product Owner can gather input from key stakeholders, SMEs, customer service, customer focus groups, their Scrum Team, surveys, and so on.
- The order may change after collaboration with the Development Team, but the intent is to rank the list from highest value items at the top to lowest value items at the bottom so time is not wasted on low value items during Release Planning.
- This ordered Product Backlog is used to determine what Stories will be included in each Release.

This page intentionally left blank.

MoSCoW

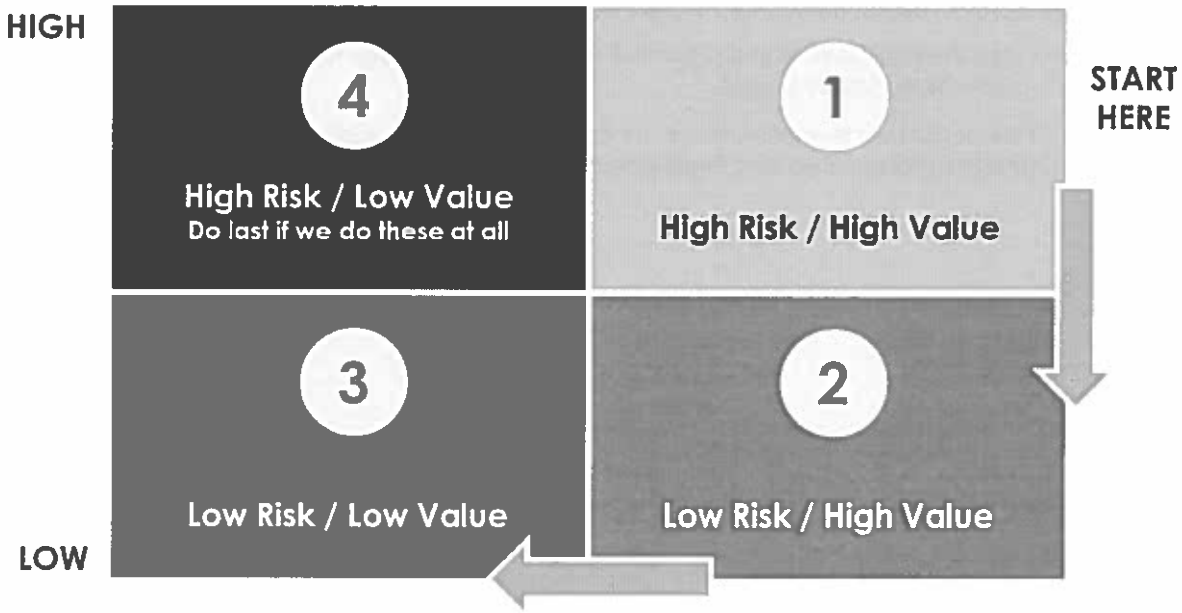
M	*MUST* have	The definition of MUST is that the feature needs to be included for the product to work.
S	*SHOULD* have	SHOULD have the feature if at all possible, or the more of these features the better but the product will still work without them.
C	*COULD* have	COULD have the feature if it does not affect anything else and there is time permitting.
W	*WOULD* have	Will not have the feature this time but WOULD like to include it in the future (these will go back to the Backlog or stay in the Backlog for another Release).

Source: Dynamic System Development Method

- The MoSCoW technique for ordering the Product Backlog comes from an early Agile approach, Dynamic System Development Method (DSDM). MoSCoW has been widely adopted in the Agile community.
- In order to plan a Release, the Product Owner must prioritize items on the Backlog. The goal is to ensure that the highest priority items, or the "MUST haves" and the "SHOULD haves" make it into the Release. Lower valued items, or "COULD haves" could make it in the Release if there is time. "WOULD haves" are items that do not make the cut, but they remain on the Backlog for a future Release.

Ordering Based on Risk & Value

Consider risk and value as you order.



Source: Cohn, Mike. *Agile Estimating & Planning*

- It's important to consider risk and value as you order the Backlog.
- Start by looking for Stories that are high risk and high value and assign them a 1. Addressing high risk/high value items early increases the chance of working through the complexities involved with high risk items and still meeting the Release date.
- After high risk/high value items we focus on those high value items that are low risk since these are the easy wins.
- Finally, if there is time, we address low risk/low value items.
- Items that are high risk and low value may not be addressed because they are not necessarily a good investment or good use of time.

Exercise: Order the Product Backlog

Instructions:

- In your Scrum Teams, order your Product Backlog:
  - Use the MoSCoW technique or the Risk-Value technique to rank the list from highest to lowest.
- If your Product Owner is in attendance, he or she should drive this since the Backlog is one of their primary responsibilities and feeds directly into Release Planning.

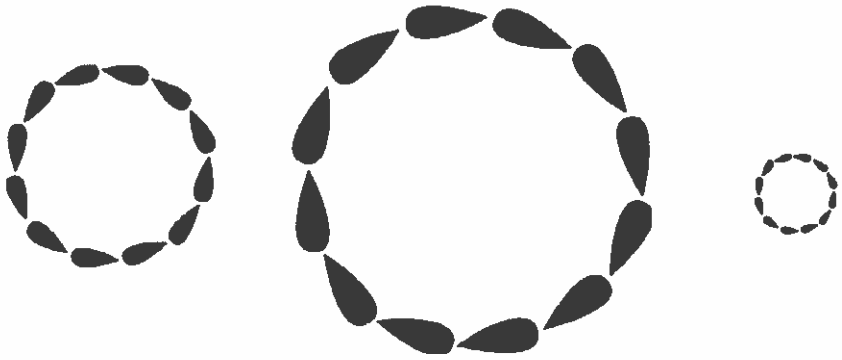
Assessment: Manage the Product Backlog

Questions:

1. True or False? The Product Owner orders the Product Backlog based on business value.
2. True of False? MoSCoW is one method for ordering the Product Backlog.

Estimating

Relative Estimation



Which circle is the biggest?

- Which circle on this slide is the biggest?
- How did you reach that conclusion? If you compared the three, that is exactly the premise behind relative estimating.
- An estimate is really a best guess. Traditional estimates have been in ideal hours or days and rarely tend to be accurate. Therefore, most Agile approaches use some form of relative estimation.
- Relative estimates are not time-based (days, hours, weeks); instead, they are based on the effort, risk, and complexity of each item as compared to others.
    - We compare the item to several others to determine its size in relation to the other items.
- Traditional estimates have been in ideal hours or ideal days and rarely tend to be accurate.
- The increased amount of time spent on estimating in hours or days does not statistically produce a more accurate estimate.

Accuracy or Consistency?

To establish consistency the Development Team must:

- Use a consistent approach to estimation.
- Establish a baseline.
- Remain stable.

- With relative estimation, consistency is more important than accuracy. Consistency can lead to predictability, which is what we encourage Teams to strive for with agility.
- If the Development Team can consistently complete a specific amount of work in a given Sprint, we can predict when we can deliver future Backlog Items or Releases.
- This requires the Development Team to:
  - Use a consistent approach to estimation.
  - Establish a baseline.
  - Remain stable. If an organization constantly swaps out Development Team members, the baseline changes and the opportunity for consistency is lost.
- On the next few slides, we will take a look at some common estimation techniques.
- As a self-organized, empowered Scrum Team, you should agree with each other on the best method that will work for the problem you are trying to solve, for your Team, and for your organization.

Discussion: Estimation

Questions:

- Why do we estimate?
- What has been your experience with estimating?
- Who does the estimation in your organization?
- And what can be done to improve estimating in your organization?

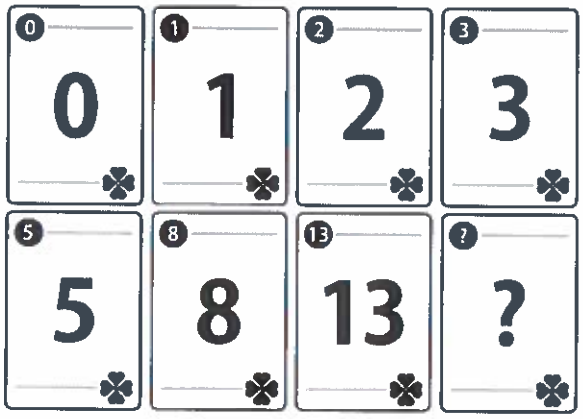
Story Points

Story Points are an arbitrary value assigned to User Stories by the Development Team to represent the effort required to complete the Story.

Source: *Agile Estimating & Planning* by Mike Cohn

- Story Points are an arbitrary value assigned to User Stories by the Development Team to represent the effort required to complete a User Story. It tells the Team how hard the Story is based on complexity, unknowns, or how much of it there is and relative to other Stories:
  - In most cases, the Story Point range is a specific series of numbers or words (2, 3, 5, 8, 13) or X-Small, Small, Medium, Large, X-Large.
  - The Team starts by identifying a baseline Story they all relate to.
  - Then, they size the rest of the Stories by comparing them to the baseline Story.
- If your Scrum Team is interested in adopting the use of Story Points, it is important to note that there is no conversion between Story Points and time.
- Story Points do not equal hours, days, or weeks. They are intentionally nebulous so the Development Teams can focus on the complexity of effort for a Story relative to other Stories.

Planning Poker®



- Planning Poker® is a consensus-based estimating technique. It is a fun and effective card-based approach.
- The Scrum Master facilitates.
- The Product Owner describes the Story or feature and answers questions the Dev Team may have in order to size the Backlog Item.
- Size is measured in Story Points. The Team uses the cards to vote on the size.
- The Development Team has the final say on all estimates.

Planning Poker is a registered trademark of Mountain Goat Software, LLC

- Planning Poker® is a registered trademark of Mountain Goat Software, LLC.
- This is a consensus-based estimating technique, similar to what is described in the Project Management Institute's Project Management Body of Knowledge when they refer to "Wideband Delphi."
- It can be useful, however, as it invites participation from the WHOLE Development Team.
- Those traditionally focused on development may not agree on an effort. Discussing viewpoints with each other can lead to consensus. When those who are thinking about Acceptance Criteria and testing weigh in, another level of complexity can be exposed if particular data is needed to test, or a particular environment, etc.

How to Play Planning Poker®

Steps:

Step 1	Each estimator has a deck of Planning Poker® cards.
Step 2	To establish a baseline, the Development Team identifies a relatively small Story and assigns it a value of 2.
Step 3	The Product Owner describes a Story and allows for brief discussion.
Step 4	Each estimator selects a numbered card from the deck for their estimate and places the card face (number) down.
Step 5	All cards are shown at the same time once the moderator (Scrum Master) gives the cue.
Step 6	The Development Team discusses differences between the estimates.
Step 7	Estimators re-estimate to reach convergence; the Development Team plays three rounds of Planning Poker® to reach consensus.

- These are the instructions for how to play Planning Poker®.
- Your instructor will provide you each with a hand of the Planning Poker® cards.
- For virtual or distributed Development Teams, you may also play "virtually" by using a session at planningpoker.com.
- Many smart phones also have free applications for download that are referred to as Planning Poker®, Agile Poker, or Agile Estimating cards.
- Most Development Teams decide on a best method for reaching convergence. One approach is to limit the number of rounds of voting.
- Some Development Teams take an average, or a mean number that they see in the group, or err on the side of taking the highest as the outlier in case risks are realized or until they mature in their process.

Class Exercise: Dog Points

Your instructor will provide direction for this exercise.

Exercise: Estimate the Product Backlog with Planning Poker

Instructions:

- In your Scrum Teams, use your Planning Poker® cards to estimate the Stories in your Product Backlog. Start at the top and record the size on each Story card or sticky note:
    - Remember the team roles and responsibilities!
- Estimate as much of the Backlog as you can, at least 3-4 items.
- In the last five minutes of the exercise:
    - Take any remaining unsized items and assign random Story Points to them. It's OK, this is just a simulation!
    - Once all items have a size, add up all the Story Points in the whole Backlog. Write this number on a sticky or card note by itself.

This page intentionally left blank.

Velocity

- Velocity is the number of Story Points a Development Team completes in a Sprint.
- At the end of each Sprint, the Story Points of all completed and accepted Stories are added up.
- A Development Team's average velocity is a running number that stabilizes over time to become fairly consistent, and becomes a number a Team can use to help with Sprint Planning.

Sprint 1 – 5+3+5+8+8+1+8=38
Sprint 2 – 5+8+3+5+8+2+3=34
Sprint 3 – 5+8+8+3+5+8+8=45
Average running velocity – (38+34+45)/3=39
This Development Team should forecast Sprints and Releases using approximately 39 Points per Sprint, plus or minus some uncertainty buffer.

- Velocity is the number of Story Points a Development Team completes in a Sprint. It's main purpose is to predict Release dates, but it can also help to create Sprint forecasts.
- You need a minimum of three to five Sprints to determine velocity you can use for planning:
  - At the end of each Sprint, add up the Story Points of all completed and accepted Stories.
  - Add the Story Point total for all the Sprints together.
  - Divide the total by the number of Sprints. This gives you the average running velocity.
  - Use the average running velocity plus or minus some uncertainty buffer to forecast future Sprints and Releases.
- In the example on the slide, the average running velocity is based on three Sprints. The Development Team should forecast Sprints and Releases using approximately 39 points per Sprint, plus or minus their uncertainty buffer.
- Each Team's velocity will be entirely different because each Team will have a different baseline of Story Pointing:
  - One Team might have a velocity of 20 while another Team's velocity is 100.
  - Both Teams are delivering high-quality Stories for their respective Sprints and working the same hours.

Exercise: Forecast a "Scope Fixed Release" Using Velocity

Scenario: Your boss comes to you and says, "How long will it take for you to deliver all of the items in the current Product Backlog?" She says she wants that number in weeks.

Instructions:

1. Your instructor will give you an estimated velocity to forecast with. Assume 2-week Sprints.
2. Number of Sprints = total Story Points in Backlog/forecasted velocity. Then convert this into weeks.
3. Now add and subtract 20% to give a range to buffer for some uncertainty.
4. Be prepared to tell your boss the estimate of how many weeks it will take to deliver this scope.

"T-Shirt Sizing" Estimation

- This technique helps Development Teams group Stories together for comparison and then group similarly sized Stories.
- To avoid confusion between the numbers in Points and the temptation to correlate to hours, days, or weeks, this technique uses "t-shirt sizes": small, medium, large, and so on.

- Another technique that has been gaining popularity in the Agile community is an Affinity Based Estimation technique using "t-shirt" sizes.
- This helps Development Teams break away from ideal hours or days and truly focus on how complex something is compared to something else.
- This technique involves comparing Stories to each other but then grouping "like" Stories together under the buckets of extra small, small, medium, large, and extra large.
- This technique can be relatively silent if you have the Story cards written or printed out.
- Development Teams can discuss after the "shuffling" has stopped and everyone has made their way through the pile of Story cards.

Assessment: Estimating

Questions:

1. True or False? Relative estimates are time-based.
2. True or False? When you use Story Pointing to estimate, you assign points to a baseline Story and then size the rest of the Stories by comparing them to the baseline.
3. True or False? A Development Team's average velocity is a running number that stabilizes over time and can be used to help with Sprint planning.

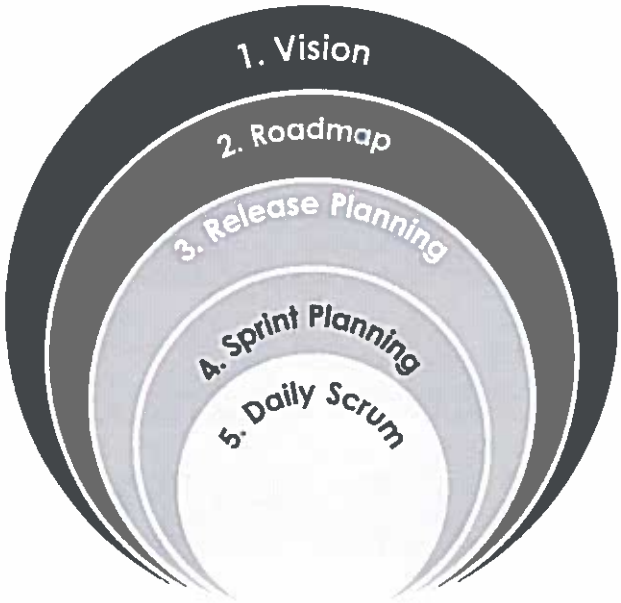
TEKsystems
Our people make IT possible.

Agile Planning: Release Planning

Agile Planning – Release Planning

Release Planning:

- Is deciding what to build and in what order.
- Involves planning multiple Sprints or Iterations to predict when a Release (or Releases) might be delivered.
- Occurs as often as needed to build the foundation on which to deliver the Vision.



- The Product Backlog with our requirements and Stories is an input to holding Release Planning.
- Release Planning gives direction to the project. It helps us decide what to build and in what order and answers the questions:
  - How many Sprints will it take us to deliver this set of features.
  - How much of this scope can we get done by a specific date?
  - How many people or Teams do we need for the project?
- The Release Planning session gives us the opportunity to see how many features or which of the features on our customer's list can be delivered by a particular date given a particular budget and/or Development Team. If all of the features are required, the Release Planning session can be used to determine the date when all of those features can be delivered with the given Scrum Team.
- Release Planning can occur at any time that a new project is kicking off.
- Some organizations standardize on their Releases to production (e.g., once per quarter) in which case Release Planning can fall in sync with that schedule.
- Release Planning is a significant session. Many who do quarterly Releases invest a full 6-7 hour Release Planning session to frame up the Release. It can take less time for more frequent Releases with a smaller feature list or Product Backlog.

This page intentionally left blank.

Release Planning Readiness

Key inputs to Release Planning are:

- Vision and Roadmap.
- Product Backlog ordered by business value.
- Scrum Team working agreements on Estimation approach.
- Velocity, if the Scrum Team has worked together on a previous Agile project.
- If the above items are not available, the Scrum Team is not ready for Release Planning.
- Scrum Masters should coach the Product Owner and Scrum Team to prepare for this meeting.

- The Product Owner facilitates the Release Planning session. Release Planning cannot take place if the Product Owner is not ready and has no ordered Product Backlog.
- The Scrum Master may coach the Product Owner on getting ready for this session or co-facilitate to keep it moving.
- Key Inputs to Release Planning are:
  - Vision and Roadmap.
  - Product Backlog ordered by business value.
  - Scrum Team working agreements on estimation approach.
  - Velocity, if the Team has worked together on a previous Agile project.
- If these items are not available, the Scrum Team is not ready for Release Planning.

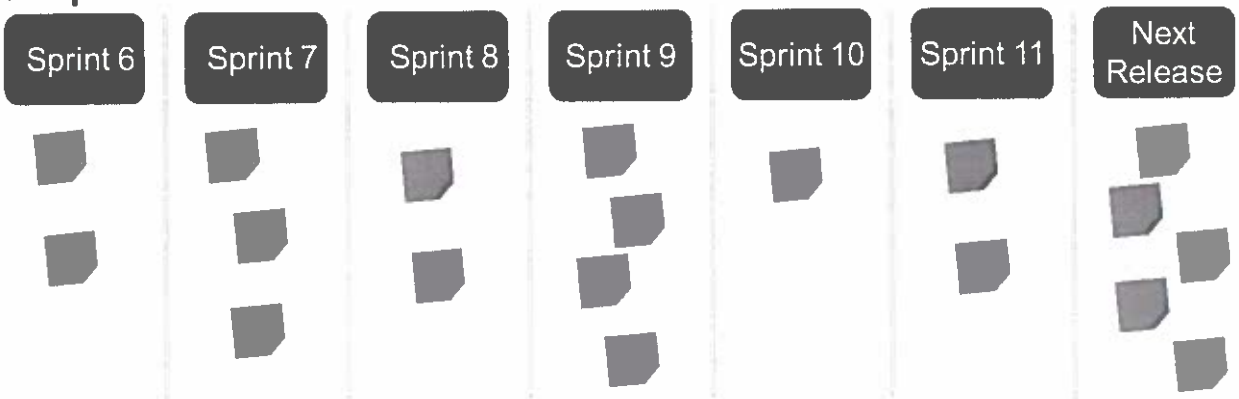
Release Planning Meeting

To hold Release Planning:

- The Product Owner describes and facilitates discussion on each Story or Product Backlog Item.
- The Development Team has a timebox for discussion – typically five minutes per Story.
- At the end of the Discussion Timebox, the Development Team estimates.
- The Scrum Master Timeboxes everything to keep the meeting on track.

- If all the inputs are available, we are ready to hold Release Planning.
- This is not a tasking session, nor is it a detailed design session.
- It is a high-level planning session in which we are discussing just enough detail to provide rough order-of-magnitude sizing.
- The Product Owner typically reads or explains a Product Backlog Item or User Story.
- The Development Team is given a timebox for discussion—typically about 5 minutes per Story.
- When the time is up, the Development Team estimates the work based on complexity and effort relative to similar work:
    - They should have agreement on what their "baseline" is.
    - What's the definition of "Small," for example, or "5 points"?
    - If they have Stories identified to continue to compare back to, this process will be much more efficient.
- Scrum Masters help keep this whole thing moving and ensure that all of the players are prepared.
- There is no prescription for who captures information in this session but it does need to be captured. Acceptance Criteria that come out of discussion, the resulting estimate in Points or t-shirt size, etc.

Sample Release Plan



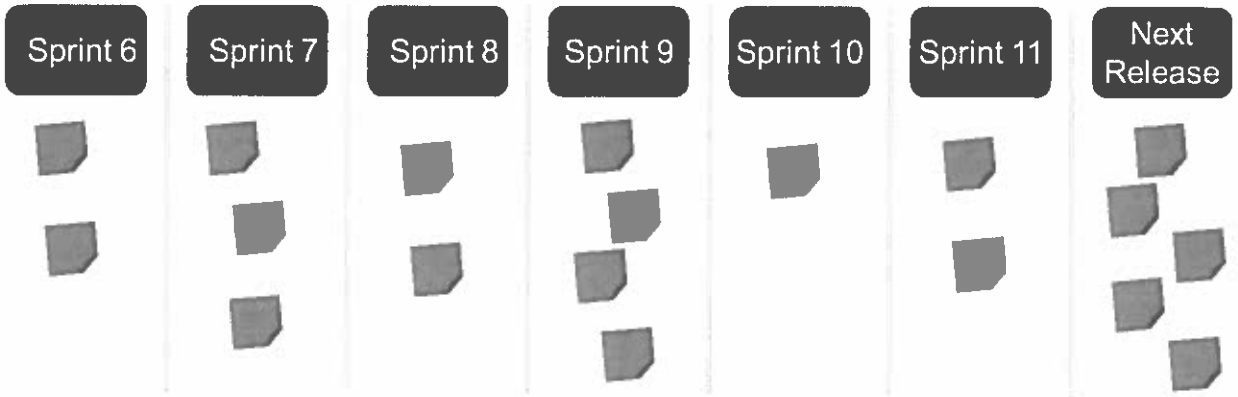
- A Release Planning session is not a large effort and does not need to be a large effort.
- Shows order of Stories, approximately where they will fall in Sprints.
- Allows Team to focus on highest ordered items, try to get them completed by end of Sprint, then shift to the next in order of the Product Backlog.
- Always subject to change based on reordering, results/actual progress, changing biz conditions.
- Updated after each Sprint to reflect reality as it emerges, as described above.

- This is an example of a Release Plan.
- It is not a large plan such as a Microsoft Project plan.
- It's enough to give visibility to the order of the Stories and approximately where they fall in the Releases' Sprints.
- We don't want to waste time on items in the Sprint that begins on July 5 now in May because something in our business may change and the Product Owner may take that item off the plate and replace it with something else.
- This type of planning allows the Development Team to focus on what is highest priority, and to get it done by the end of the Sprint, and then shift focus to what is next in priority.
- The Product Owner always needs to be refining the Backlog and staying ahead of the Development Team, so that when it is the next Sprint Planning time, he or she can ask the Development Team to work on only the next highest priority items and not waste time/cycles.

Exercise: Release Planning Meeting ("Date Fixed Release")

Instructions:

- In your Scrum Teams, hold a Release Planning session. Assume that the Release period will be six 2-week Sprints (3 months), and culminate in a single Release at the end of the 3 months.
- On the wall or a table, lay out seven sticky notes (or cards) horizontally, one for each Sprint ("Sprint 1," "Sprint 2," etc.), and then the seventh one will say "Next Release" (meaning that Story will not likely make the Release).
- Using your forecasted velocity per Sprint, place Stories into Sprints (columns) by placing the Story under the Sprint it will go in based on your forecast.
- Did you have enough Stories to fill the Release period? Did you have too few?



This page intentionally left blank.

Assessment: Agile Planning: Release Planning

Questions:

Product Owner

1. True or False? The Scrum Master describes and facilitates discussion on each Story or Product Backlog Item during Release Planning.
2. Which of these are key inputs to Release Planning? Select all that apply.
   A. Vision and Roadmap
   B. Product Backlog
   C. Scrum Team working agreement on estimation approach
   D. Velocity if the Team has worked together

The Sprint

Daniel Pink
Drive book

Practology - Why people do what they do.

Sprint Planning Readiness

Key inputs to Sprint Planning are:

- The Release Plan.
- The ordered, estimated Product Backlog.
- The Scrum Team's upcoming capacity for the current Sprint, including any known time off, appointments, etc.
- The Scrum Team's forecasted velocity, if they have been together for a previous Sprint or Sprints.
- The company calendar, including any holidays, town hall meetings, etc.

- In order to prepare for Sprint Planning, we need the Release Plan and to ensure that the ordered Product Backlog has the items at the top in a ready state to work on.
- The Product Owner should have included the necessary detail and be prepared to discuss with the Development Team at Sprint Planning.
- Any known company holidays or meetings should be brought to planning, as well as any Development Team member vacations, appointments, etc.
- If a Development Team has been together for several Sprints and knows their velocity, this is a key planning metric that we will use as input into planning.

Capacity vs. Velocity

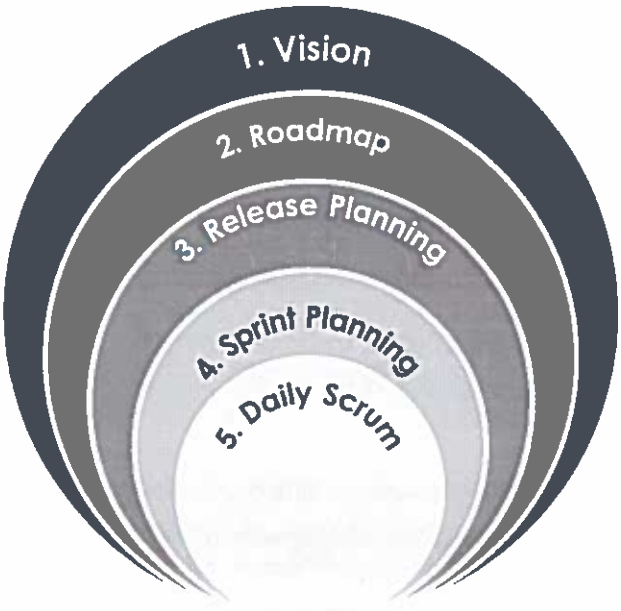
- Capacity is *how much time the Development Team has available* to work during the Sprint, including the known holidays, time off, etc.
- Velocity is the *amount of Product Backlog that the Development Team is historically capable of completing in a Sprint,* based on empirical data—it is a planning metric.
- It is important for the Sprint length to remain consistent for the Development Team to establish a cadence and forecast velocity.
- When Sprint lengths or Development Team staff change throughout the Release, this heavily disrupts the usefulness of velocity to use as input for Sprint forecasting/planning—remember that and adjust velocity forecasts as needed.

- Capacity is how many hours during the Sprint the Development Team is available to work.
- This involves looking ahead at the next 2 weeks, for example, if we are doing 2-week Sprints, and taking into account those company holidays, vacations, appointments, etc.
- Some Development Teams assume that there are 6 productive hours in a day as a guideline, not 8, accounting for any meetings, answering emails, etc.
- Others assume 7.
- It will vary from company to company.
- Capacity is NOT the same as velocity.
- Velocity is the amount of Story Points or work that the Development Team is historically capable of producing in a Sprint.
- This number is a "rolling average."
- For example, if our velocity in Story Points for Sprint 1 was 60, Sprint 2 was 73, and Sprint 3 was 68, our average velocity going into Sprint 4 will be 67.
- In other words, the Development Team should take on no more than 67 Story Points worth of work for Sprint 4.

Agile Planning – Sprint Planning

Sprint Planning

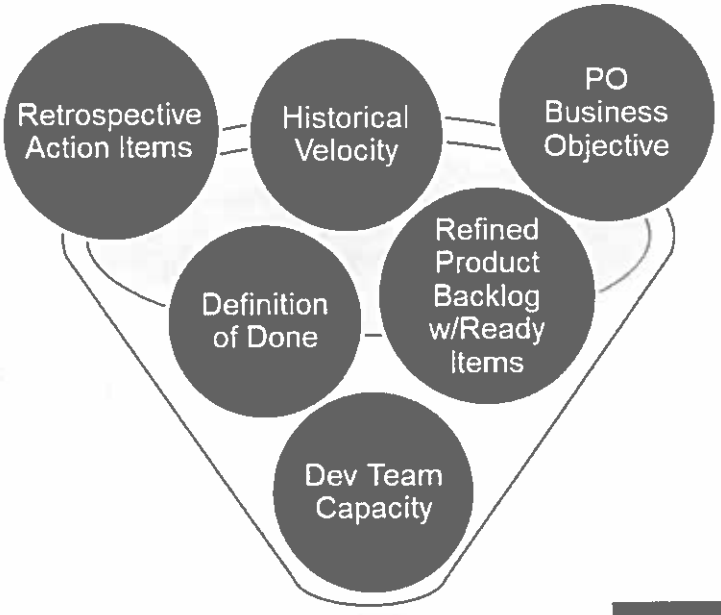
- Also known as an Iteration in some approaches, it occurs every 1-4 weeks depending on the Team or organization.
- A Sprint is the Timebox in which the Team completes a working product increment.
- Scrum Teams maintain a "cadence" to determine a consistent pace and baseline so they can estimate how much work can be completed per Sprint.
- This allows the Scrum Team to consistently estimate how much work can be completed.



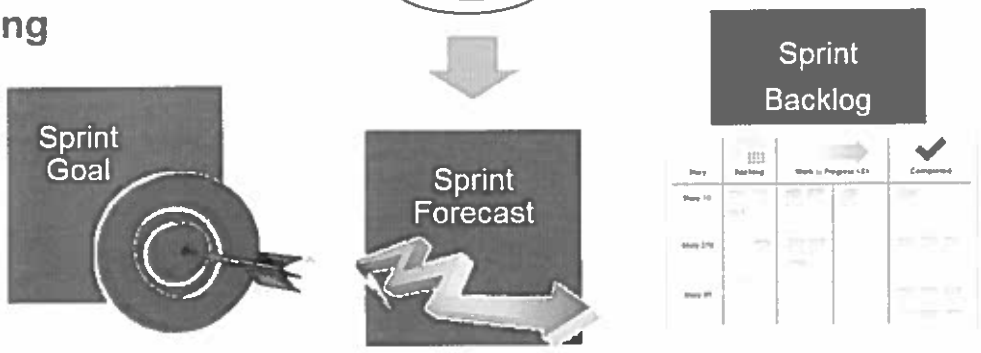
- Let's level-set where we are at with Agile Planning. We've got a product or project Vision and a high-level Roadmap.
- We've simulated a small slice of Release Planning so that we understand what items will be at the top of the Backlog.
- What's highest on the list will get addressed in the first Sprint Planning session.
- We're now ready to take a look at Sprint Planning.
- If your Team has been together working with Agile for awhile, you may have already adopted a rhythm or cadence.
- It is important to keep this Sprint timebox (1 week, 2 weeks, etc.) consistent.
- We would not do a 1-week Sprint and then say "extend" the Sprint to 8 days next time or change that Timebox.
- It takes away any baseline and any opportunity for the Team to be consistent and become predictable in what can be achieved in a Sprint.

The Sprint Planning Event

Inputs to Sprint Planning



Outputs from Sprint Planning

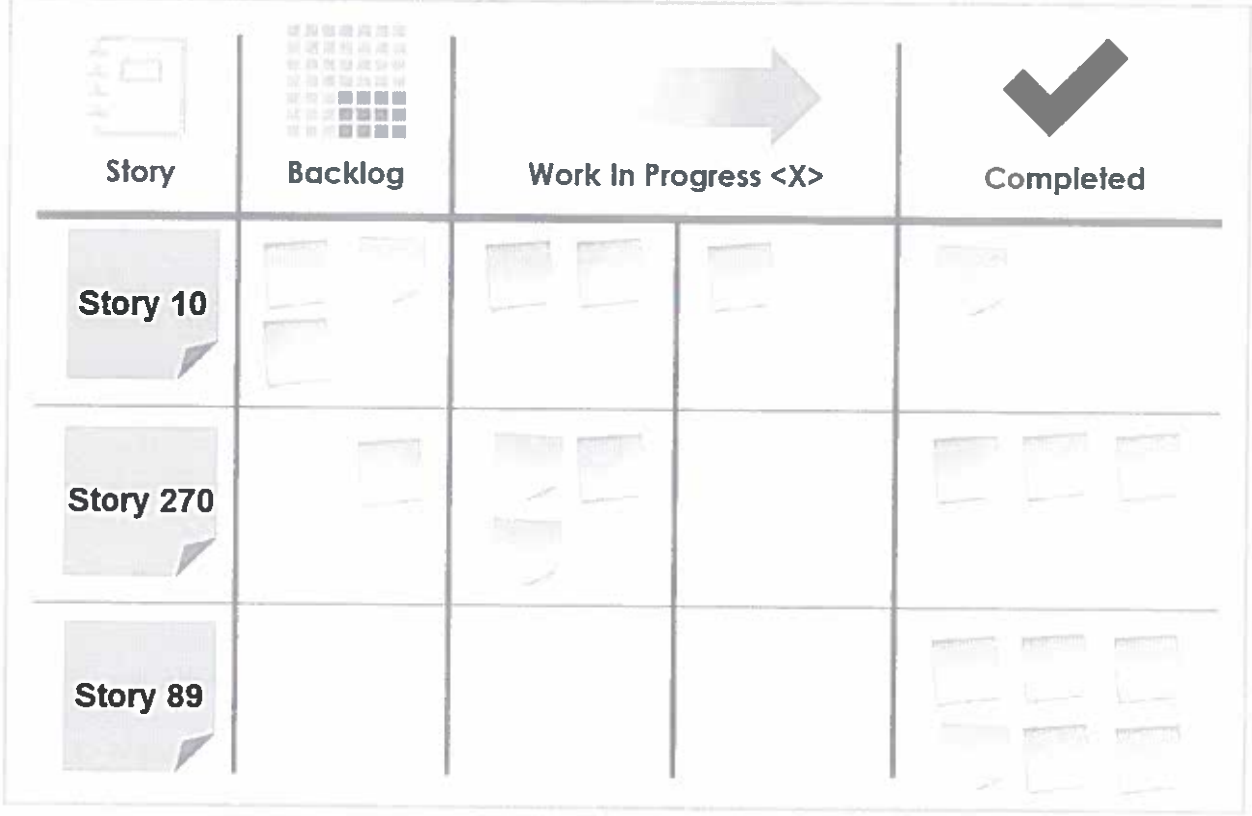


- The objective is for the Dev Team to realistically forecast the Stories and tasks that will get them to completion within the Timebox (Sprint).
- The Product Owner is present to answer questions about User Stories or Product Backlog Items.
- The Scrum Master facilitates as wanted or as needed, to keep the event moving.
- Often, the Dev Team decomposes (breaks down) the Product Backlog Items into tasks that they will complete during the Sprint, and estimates those tasks. The tasks are often broken down to units of one day or less.
- Sprint Planning is a working session for the Development Team, the Scrum Master, and the Product Owner.
- It is NOT a management or an executive forum.
- The Product Owner is there to clarify Backlog order and User Story requirement details.
- The Stories should have been previously sized in Release Planning or Product Backlog Refinement.

Sprint Planning Output: The Sprint Backlog

The Sprint Backlog:

- Is a set of Product Backlog Items selected for the Sprint, plus a plan for delivering the product increment, and realizing the Sprint Goal.
- Often expressed as a set of Stories that are broken down into tasks, as shown below.
- Is a forecast of what the Dev Team thinks they can complete.



- The Sprint Backlog is a set of Product Backlog Items selected for the Sprint, plus a plan for delivering the product increment and realizing the Sprint Goal.
- The Sprint Backlog is a forecast by the Dev Team about what functionality will be in the next increment, and the work needed to deliver that functionality.
- Defines the work the Dev Team will perform (in tasks) to turn the Product Backlog Item into a "Done" increment.
- Should be visible and include all work for the Sprint, with enough detail for work in progress to be understood at the Daily Scrum.
- Some Teams break Stories into tasks one at a time, while other Teams will work in small groups to break Stories down in parallel—to speed up Sprint Planning.

- Some Development Teams will be tempted to go over their velocity if they see that hours are available in their capacity—this should be avoided, and room for uncertainty should be built in.
- If they overload their plate, they take away the flexibility that the framework provides for them to inspect and adapt.

Exercise: Sprint Planning

Instructions:

- In your Scrum Teams, simulate Sprint Planning. Note that your Sprints will be made up of three simulated days. Each day is 10 minutes long.
- Discuss velocity and capacity, and review the Product Backlog.
- Break the first Item on the Backlog down into tasks, adding it to the Sprint Backlog. Repeat as you have time.
- Product Owners answer questions, Scrum Masters keep the process moving, Dev Team members break Stories down into tasks and estimate tasks.

During the Sprint

Product Owner

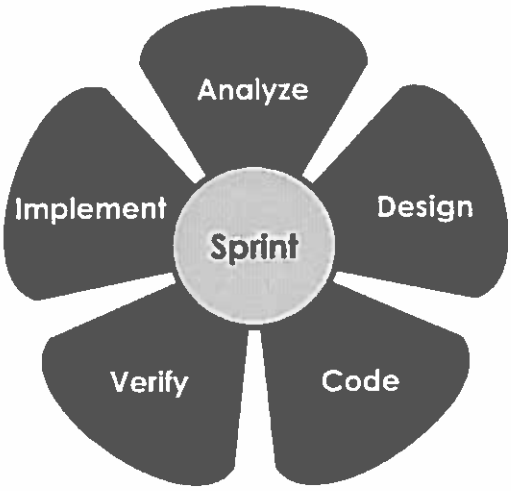
- Clarifies requirements.
- Gives feedback (over the shoulder check):
  - Accepts/rejects.
  - Recommends tweaks.
  - Product Backlog Refinement.

Dev Team

- Swarms/pairs on items.
- Presents completed PBIs to Product Owner.
- Updates Sprint Backlog.
- Updates Sprint Burndown.
- Product Backlog Refinement.

Scrum Master

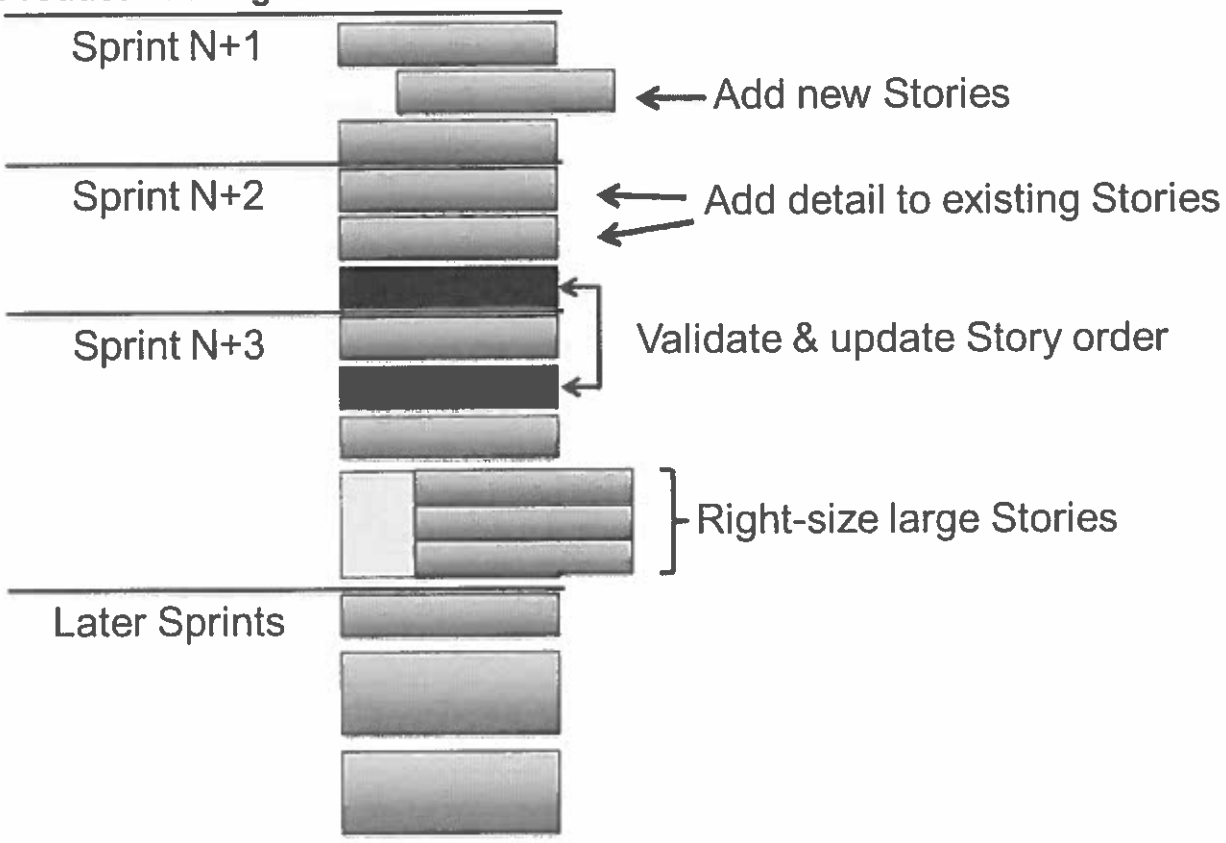
- Removes impediments.
- Teaches/coaches Team.
- Teaches/coaches org.



- Before we simulate the Sprint, let's take a look at what typically happens during a Sprint.
- During the Sprint, the Dev Team is working on Story tasks, swarming and/or pairing as necessary.
- The Dev Team seeks clarification from the Product Owner as needed.
- The Dev Team collaborates, communicates, updates the Sprint Backlog (Scrum Board) and burndown chart, and inspects and adapts at the daily Scrum.
- During the Sprint, as soon as the Dev Team is finished with a Backlog Item, they will present it to the Product Owner for an "over the shoulder check" to accept, reject, or suggest small tweaks.
- During the Sprint, the Scrum Master removes impediments that the Dev Team cannot remove, and coaches and teaches the Team and the organization on how to do Scrum well and how to benefit the most from Scrum's strategic benefits.
- When people are new to adopting Agile, one thing that they tend to gravitate toward is the meetings identified, because it's an area of comfort from a traditional process.
- There is a common occurrence for any of these meetings to run far beyond their intended length because Development Team members, Product Owners, and maybe even some new Scrum Masters are afraid it is one of the only times they will talk so all details must be fleshed out to resemble traditional project requirement documents, design documents, etc. This is not the case.
- Scrum prescribes Timeboxes for each of the Scrum events to discourage Teams from spending too much time in these events.

- Everyone is encouraged to collaborate all the time during the Sprint.
- This is why the emphasis is on co-location.
- If co-location is not possible, what technology is available to promote collaboration?

Product Backlog Refinement

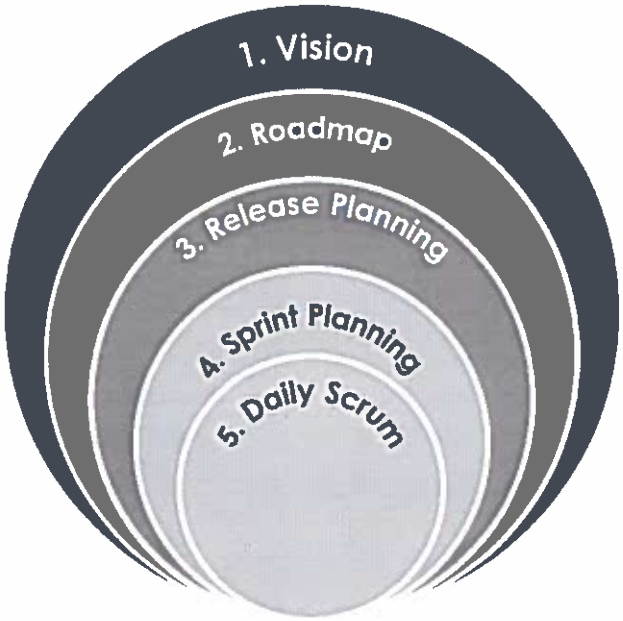


- The intention of Product Backlog Refinement is:
  - To identify and create new Stories.
  - To add detail to existing Stories.
  - To split large Stories into right-sized Stories.
  - To validate and update ordering.
  - To prepare for Sprint Planning.
  - To prepare for Releases and Release Planning.
- Backlog Refinement is continuous.
- Refinement is about the Stories. Stakeholders help the Product Owner create the Vision, which is then translated into Stories. Lead Technical Team members (architects) and Lead Test Team members sometimes are helpful in breaking down Stories and identifying gaps and dependencies for which the Product Owners might need help identifying.
- It is good practice to gather external inputs regularly.

The Daily Scrum Event

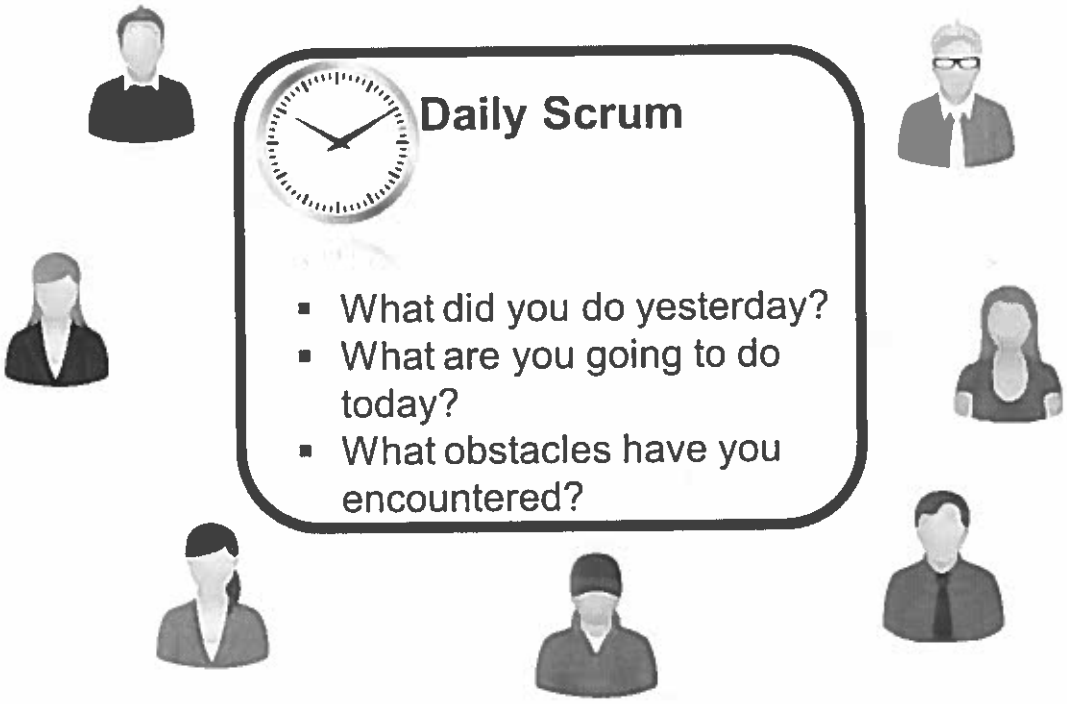
Daily Scrum:

- Daily Planning occurs at the Daily Scrum.
- **This is not a status meeting**, it is an inspect and adapt mechanism for the Dev Team.
- The Dev Team talks to each other about:
  - What they have completed.
  - What they are planning for that day.
  - Any challenges or impediments keeping them from achieving the Sprint Goal.
- This daily synchronization allows the Dev Team to make real-time course corrections to meet the Sprint Goal.



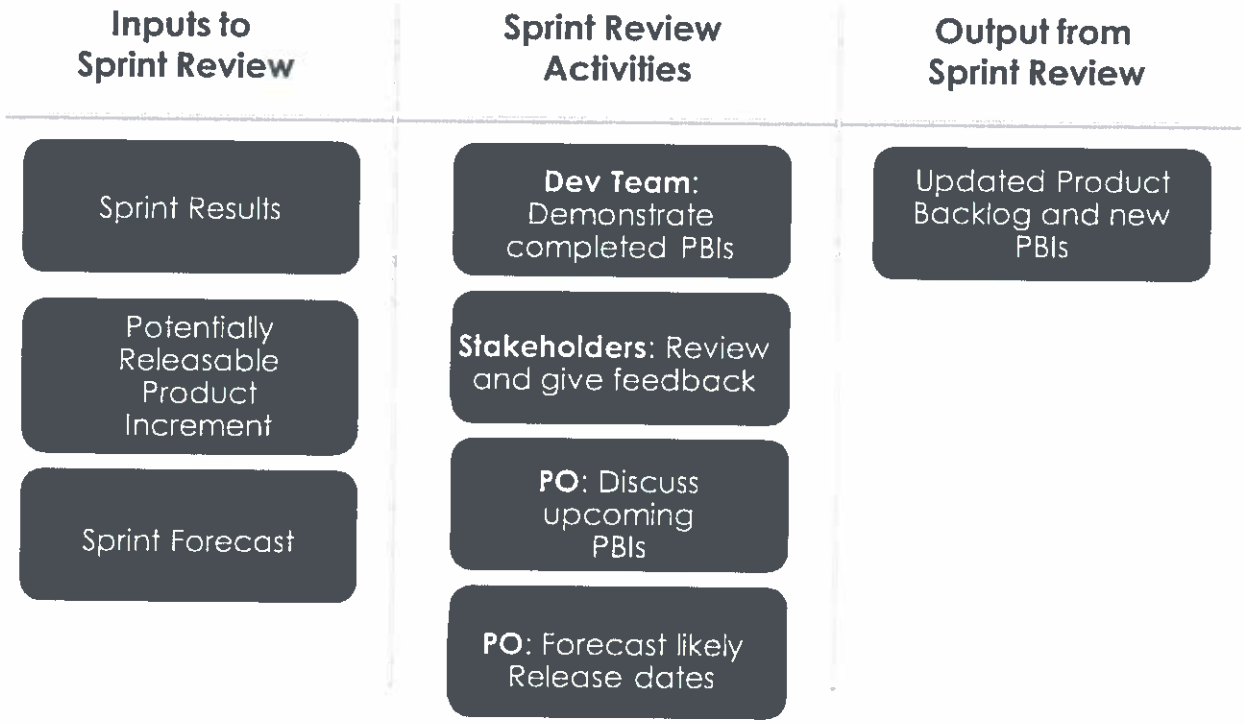
- The final layer of planning in Agile is at the daily level.
- In Scrum, we refer to this daily 'huddle" as the Daily Scrum.
- Other methods will refer to it as Daily Stand Up.
- Yes, participants are expected to stand up.
- The reason that we stand at this daily meeting is so that the 15-minute Timebox is respected.
- This is not a status meeting to the Scrum Master, the Product Owner, or management and it should not turn into one.
- The Development Team talks to each other.
- Again, this is not an executive or management forum but an opportunity for Development Team members to sync with each other on: What they accomplished yesterday, what they will focus on today, and what is blocking them or impeding them from getting to "done" or making progress.
- If a Development Team member raises an issue that someone else can help with, the conversation does not derail into the details about this.
- The Development Team member who can help may say, "Let's chat after the Scrum, I can help you with that."
- If the Development Team needs time with the Product Owner they may raise that but they don't have that detailed conversation at Stand Up; they ask if the Product Owner can come to the Team room or meet with them right after.

Daily Scrum



- This is an inspect and adapt mechanism for the Dev Team.
- This is NOT an executive forum or a meeting to provide status to a Project Manager or the Scrum Master.
- It is ideal for the Dev Team to meet all together, physically or visually (via video conference).
- The Scrum Master and Product Owner are completely optional at this event, though they often will attend as silent observers.
- Sometimes the Scrum Master and Product Owner will communicate with the Dev Team just after the Daily Scrum (sometimes known as the "After Meeting" or the "After Party").
- Some Development Teams meet around their Sprint Backlog/task board, and this is fine so long as they don't spend most of their Daily Scrum "updating" the Sprint Backlog (turns the Daily Scrum into a waterfall status meeting).
- The Daily Scrum is held at the same time and place every day, to reduce complexity.
- If someone is always late or always misses it, the Development Team should have an honest and open conversation about this and talk about what would be the best time for everyone to make it.
- This is a forum for the Development Team to be accountable to themselves and to each other.
- Ideally, this is held by the Development Team's task board so that Team members can move their own sticky notes, update them, speak to them, etc.

Sprint Review

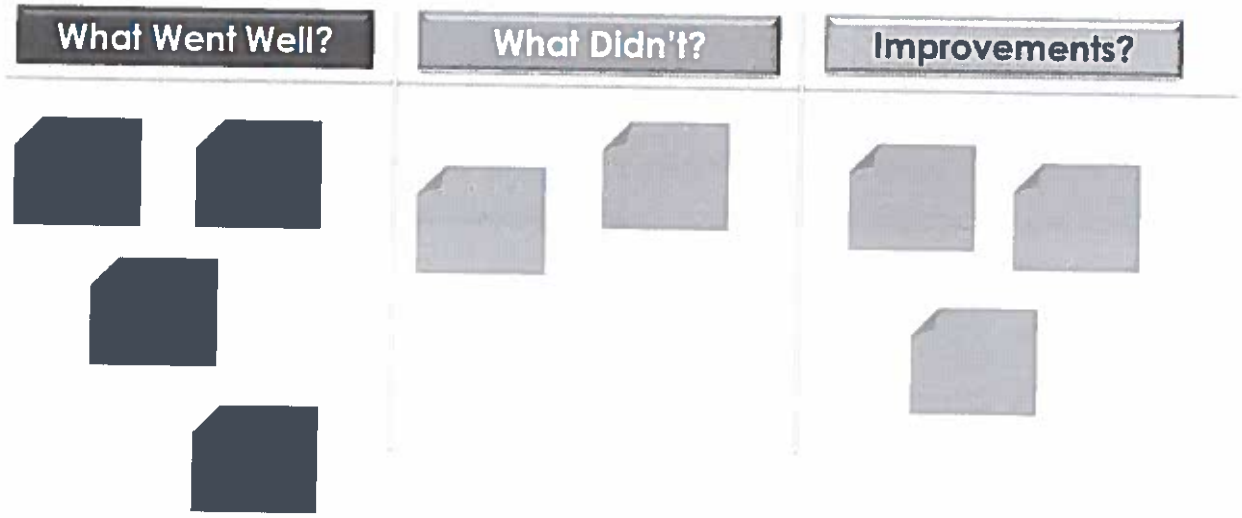


- The Dev Team demonstrates (only) completed Product Backlog Items during the Sprint Review to the stakeholders.
- Any incomplete items go back into the Product Backlog, where the PO can choose to carry them over or reorder them as needed.
- Feedback is captured in new Product Backlog Items or User Stories, and added to the Product Backlog as necessary.
- The Review should not be a surprise to anyone; preparing for this demonstration of a working product increment should be taken into account during the Sprint.
- The Sprint Review is not the first time we ask the Product Owner for feedback or have demonstrated the working product. That should occur as needed throughout the Sprint.
- The Product Owner should be the lead facilitator in the discussion with the key stakeholders in Sprint Reviews, discussing the present state of the Product Backlog and what will be worked on in the coming Sprints.
- The Product Owner discusses the progress toward milestones/Releases, and forecasts likely completion/Release dates, in a communication for key stakeholders. A lot of Teams forget to do this.
- The Scrum Master has no role other than to teach and coach how to hold a Sprint Review.
- It is important to understand that this does not imply this is the first and only time the Product Owner would be seeing the completed Product Backlog Items.
- He or she should have been involved in the process throughout and already had an opportunity to provide feedback so there is still time for the Team to respond to this feedback within the Sprint.

- The Product Owner invites the key stakeholders and customers.
- The Review is not a PowerPoint presentation but is a demo of working software. It also does not need to be lengthy.
- Any feedback received at the review in the way of new functionality, features, etc., will be captured by the Product Owner and ordered in the Product Backlog along with everything else.
- There should not be an assumption that feedback items will automatically be worked on at the next Sprint.
- The process is to put those ideas on the Product Backlog and evaluate at or by Sprint Planning.
- This is why it's important that the Product Owner be involved prior to the final review.

Sprint Retrospective

An inspect and adapt mechanism at the end of each Sprint that allows the **Scrum Team** to improve their effectiveness.



- What went well?
  - Celebrate good Team behavior. Recognize simple but effective things.
  - Were there any experiments to improve that went well?
  - Focus on the Scrum Team and how they work together to deliver value.
- What didn't go well?
  - Identify impediments to the Scrum Team's performance. If not removable by Team, then have the Scrum Master attack these.
  - Were there any experiments to improve that didn't go so well?
  - Try to focus on those items within the Scrum Team's control, and turn over the rest for the Scrum Master to do as part of their "Impediment Removal" responsibility.
- What will we improve?
  - Focus on practical changes, and be realistic in choosing the 1-2 improvement items the Scrum Team can actually change in the upcoming Sprint.
  - Beyond the 1-2 items, keeping a "Retrospective Backlog" or "Improvement Backlog" of good ideas and suggestions allows the Scrum Team to revisit those items for future Sprints.
  - The Scrum Team decides which items they will dedicate themselves to trying for the next Sprint. Some people call these "improvement experiments."
  - Sprint Retrospectives are worthless if nothing changes, so it's important for the Scrum Team to plan time into the coming Sprints to work on improvement items. Consider this as part of the capacity discussion in Sprint Planning.
  - The Sprint Retrospective should not just be about "who said what," "meeting minutes," or "good, bad, and ugly." The focus should be on choosing improvements to make.

  - Many Scrum Teams don't spend the time needed in Retrospectives to get to "root causes." Instead, they try to solve "surface" problems that often don't yield much improvement at all. Get to the root!

- The Scrum Master optionally facilitates this session.
- It occurs after the Sprint Review, but before the next Sprint Planning session, so the Scrum Team can inspect and adapt as necessary and "course correct" in real time.
- The Scrum Team makes improvements *within* the Scrum Framework. The Team is not allowed to make changes to the Scrum Framework itself.
- Where the Scrum Framework provides flexibility (estimating, forecasting, Product Backlog management), Teams are encouraged to experiment, inspect, and adapt their techniques, while respecting the Scrum Framework principles and practices.
- Esther Derby and Diana Larsen identified an approach for facilitating this session in "Agile Retrospectives":
  - Set the Stage.
  - Gather Data.
  - Generate Insights.
  - Decide What to Do.
  - Close the Retrospective.

Assessment: The Sprint

Questions:

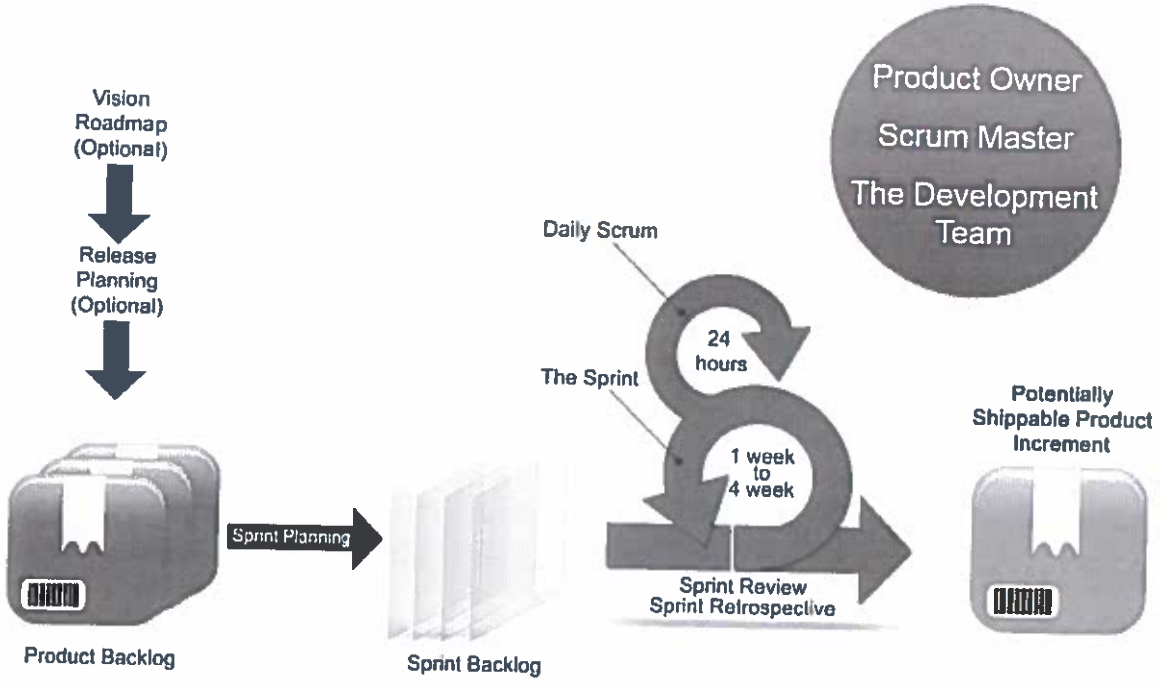
1. True or False? Outputs from the Sprint Planning event are the Sprint Goal, the Sprint Forecast, and the Sprint Backlog.
2. Capacity is how much time the Development Team has available to work during the Sprint, including known holidays, time off, etc.
   A. Velocity
   B. Capacity
3. True or False? The Sprint Review is the inspect and adapt mechanism at the end of each Sprint that allows the Scrum Team to improve their effectiveness.
4. During the Daily Scrum, the Development Team answers which questions?
   A. What did you do yesterday?
   B. What are you going to do today?
   C. What help do you need?
   D. Did you solve any problems?
   E. What obstacles have you encountered?

Retrospective

TEKsystems
Our people make IT possible.

Practice Simulations

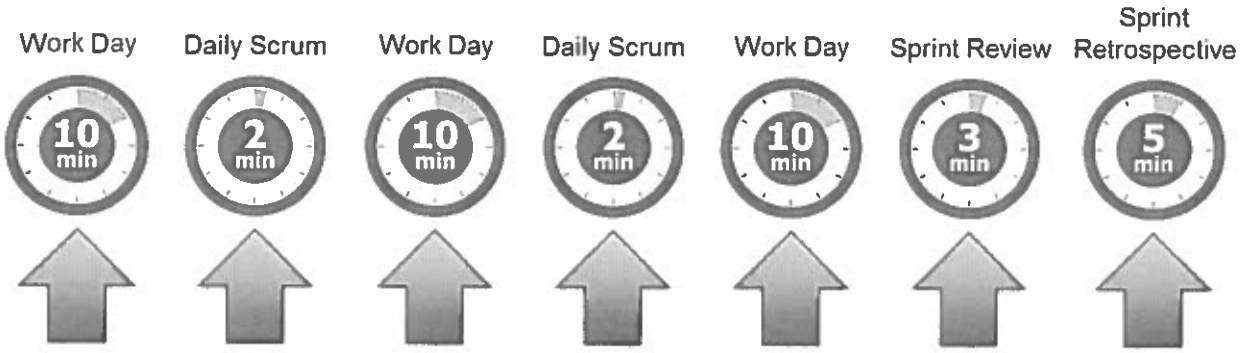
Scrum Flow



- This view provides a holistic view of the extended Scrum process.
- This may seem like a lot to understand all at one time, but this view is intended to provide the "big picture" only.
- We will be covering each of these areas in detail in this course.
- Please note that the objective of each Sprint is to deliver "working software"—even if it is an increment.
- Release Planning is an optional event and is not considered one of the "core" events in Scrum.
- Core Scrum events include: Sprint Planning, Daily Scrum, Sprint Review, and the Sprint Retrospective.
- A Potentially Releasable Product Increment that functions is still considered valuable—even if we save those increments to be released to production together after a certain number of Sprints (a.k.a., Release).

Simulation: Sprint 1

- For this Sprint simulation, follow the plan you created during Sprint Planning.
- Work on your simulated tasks as you can, creating paper prototypes.
- There will be 3 simulated "days" in each Sprint as follows:
  - Day 1: 10 minutes "work day."
  - Day 2: 2 minutes Daily Scrum, 10 minutes "work day."
  - Day 3: 2 minutes Daily Scrum, 10 minutes "work day."
  - Sprint Review: 3 minutes.
  - Sprint Retrospective: 5 minutes.



Discussion: Simulation

Questions:

- What observations do you have after the simulated Scrum events?
- Did your Team stick to a Scrum simulation or incorporate any influences from other Agile approaches?
- What challenges do you see in applying Scrum to your product or project? In your organization?

Simulation: Agile & Scrum Roles

Activity:

- We are going to move into the next round of the Simulation exercise.
- If you did not discuss what to approach differently at your Retrospective, take a moment to reflect on this.
- Did someone have to "role-play" the Product Owner or Scrum Master if they are not in attendance with your real-life Team?
- Did you have other approaches you wanted to experiment with?
- Do you have other people in your class Scrum Team who want to role-play as the Product Owner or SM?
- Take a minute to agree as a Team on any changes before we move into the Sprint 2 Planning session.

Simulation: Sprint 2

Instructions:

- In your Scrum Teams, simulate Sprint Planning, discuss velocity/capacity, and pull items from the top of the Product Backlog. Break Product Backlog Items down into tasks.
- Work on your tasks as you can, with your wireframing project.
- Following Sprint Planning, there will be 3 simulated "days" in each Sprint as follows:
    - Day 1: 10 minutes "work day."
    - Day 2: 2 minutes Daily Scrum, 10 minutes "work day."
    - Day 3: 2 minutes Daily Scrum, 10 minutes "work day."
    - Sprint Review: 3 minutes.
    - Sprint Retrospective: 5 minutes.
- Work together in breakout rooms and then be prepared to share your work at the end of the session.

Discussion: Inspect & Adapt

Question:

- How was what you did in Sprint 2 different from what you did in Sprint 1 in terms of your Scrum Team's approach?
- How did you reach the decisions to make these changes?
- Were the changes large or small?

Transparency and Information Radiators

Review: Transparency

- As a reminder, the three pillars of any empirical process are: inspection, adaptation, and transparency.
- Agile methods are empirical and allow Teams to inspect and adapt, and provide a high degree of visibility or transparency into what is happening on the project or with the product.
- The intent is that this visibility is for the Scrum Team and key stakeholders (primarily Sprint Review, Product Backlog).
- With this level of transparency, they can make "course corrections" in real time if faced with an impediment that would delay the project or the Release of business value.
- This level of transparency increases the communication on the Scrum Team and establishes trust.

Discussion: Working Product Increment

- The Product Owner or customer should realize value at the end of each Sprint.
- The Scrum Team should strive to have "production worthy" code, even if it's a product increment, and even if it will be held for Release into other increments.
- Are we carrying over unfinished work every Sprint, creating technical debt, and not reaching "done"?
- What kinds of things prevent a Scrum Team from reaching the Definition of Done?

How Google Tests Software
- good test book, light weight tests

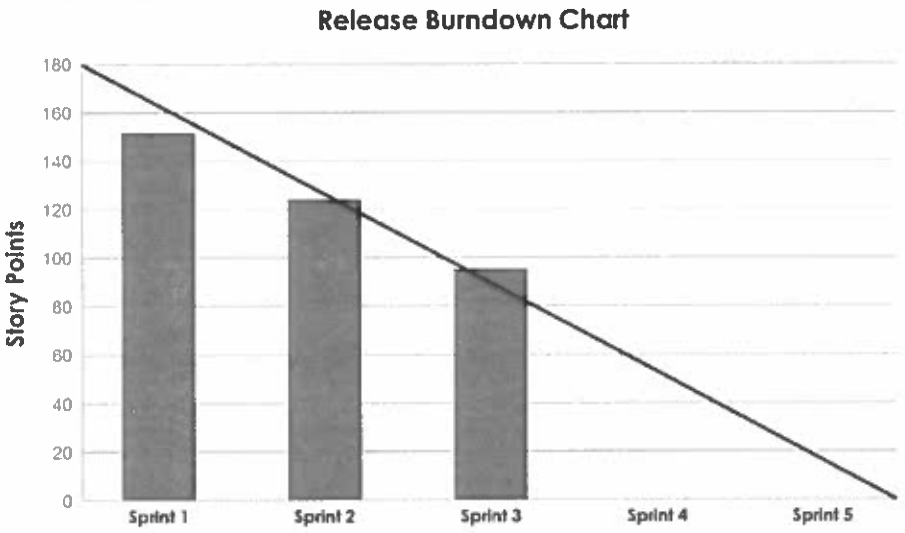
Discussion: Technical Debt

Question:

- What happens if we continually do not achieve the Definition of Done?

Release Burndown

- A simple visual of the overall completeness of a Release.
- Developed and updated by Product Owner following each Sprint.
- The baseline is the ideal line. It gives us the baseline of where we should be at any given Sprint for a Release.

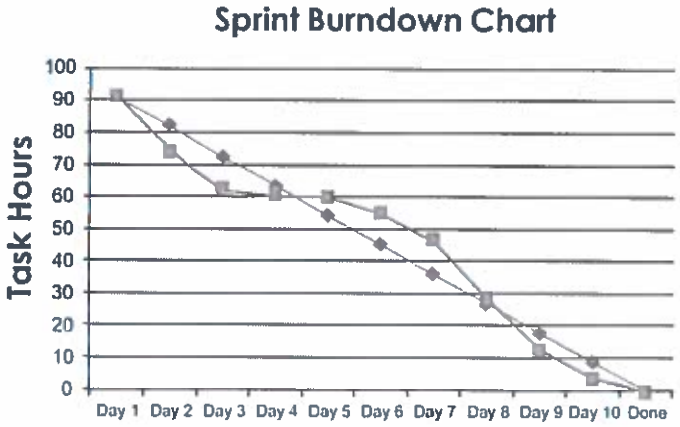


- Baseline: The black line on the chart shows where a Release's remaining work should ideally be.
- The image depicts a project with five Sprints.
- The project initially had 183 Story Points.
- The Development Team appears to be burning approximately 30 Stories per Sprint.
- This image depicts the Development Team's velocity.
- We are burning down as Sprints complete.
- At the end of each Sprint, the Scrum Master updates the completed Story Points count into the chart; the remaining work adjusts.
- The chart is a clear and simple indication of Release Health.
- Ideally at the end of the Sprint, work remaining should be zero. If all allocated Release Stories are not completed by the last Sprint in the Release, a blue bar will be present at Release five.
- If additional Stories are completed during the Sprint, then the blue bar will be present and dip below the 0 line.

Sprint Burndown Chart

Story Points

- A simple visual of the status of the Dev Team's Sprint forecast.
- Developed and updated by the Development Team every 24 hours.
- The baseline is the ideal line. It gives us the baseline of how much work the Dev Team should have remaining based on their initial forecast.
- The actual line shows how much work is actually remaining as of that day in the Sprint.



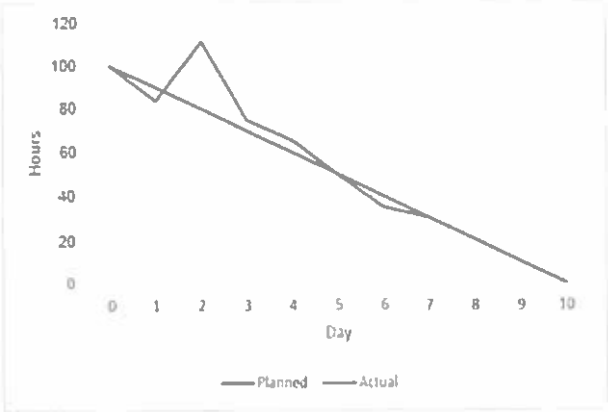
- The Sprint Burndown Chart, which is maintained by the Development Team, is an important tool.
- The Sprint Burndown Chart, if updated daily, shows the work remaining for the Development Team versus its original Sprint forecast.
- At any given time, the Development Team, Product Owner, and other stakeholders can get a feel for how the Development Team is progressing against its forecast.
- A flat line indicates no progress.
- A Spike in the line indicates a burnup of existing or newly added tasks. The Development Team made a discovery that was missed during Sprint Planning.
- A line that hits the bottom indicates there are no more hours remaining in the Sprint.

Discussion: Sprint Burndown

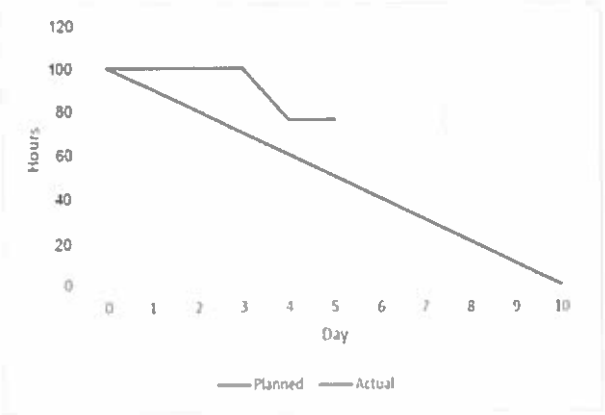
Question:

- Based on the two Sprint Burndowns below:
  - What is your analysis of the overall health of the Sprints?
  - What conversations should be taking place?

Sprint 3



Sprint 7



Assessment: Transparency and Information Radiators

Questions:

1. True or False? Transparency allows Agile Teams to inspect and adapt and make course corrections in real time if faced with an impediment that would delay the project or the Release.
2. Who develops and updates the Release burndown?
   A. Product Owner
   B. Scrum Master
   C. Development Team
3. How often is the Sprint burndown chart updated?
   A. Every day
   B. Once a week
   C. At the end of each Sprint
   D. After every Release

TEKsystems
Our people make IT possible.

Continuous Improvement

Section 19

Continuous Improvement

- Remember The Agile Manifesto principle:
  - At regular intervals, the Team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
- This is not a simple reference to a Retrospective at the end of an Iteration or Sprint.
- This refers to the concept of Continuous Improvement in any Lean or Agile process.
- The Team looks at what is and is not working with the current process, and then makes adjustments they can immediately put into practice.
- In order to continue providing value to customers and clients, it is important that we continue to re-evaluate ourselves as well.

- Most of the Agile methods or approaches have roots in Lean thinking.
- At the heart of Lean thinking is always looking for Continuous Improvement in the process that is used.
- Scrum specifically sets aside time at the end of each Sprint for the Scrum Team to inspect their process and adapt based on what is working well and what is not working very well.
- Other methods speak to Continuous Improvement by examining engineering practices for improvement (XP), increasing throughput (Kanban), etc.
- It is important to note that we're talking about improving the process, not necessarily the product.
- Product improvement will improve and quality will improve if we improve our process.

Incremental Change

- Just as we have learned about incremental product development, change can also come about incrementally.
- We have learned a lot of new values, principles, and concepts, and it would be unrealistic for an entire company to consider applying all of these at once.
- Start with a pilot Team, project, or product group to help identify where your organization could more easily adopt these practices, and where it may face the most challenges or impediments.
- When choosing a pilot to gauge how your organization will adapt to Agile, avoid high-profile or politically charged Teams, products, or projects.
- Look for a business partner or sponsor who is willing and engaged, and will fill the role of or supply a Product Owner to provide direction and feedback to the Scrum Team.

Discuss with other participants: Is your organization approaching change incrementally or attempting "big bang"?

Incremental Change (Cont.)

- Choose a Team who will give Agile a fair try:
  - Some may be skeptical.
  - Find an open-minded Team to help determine if the method is feasible within organizational constraints.
- Ensure that the project is not too big or too small.
- Projects longer than six months in duration are probably too large for an initial pilot.

- Change does not need to happen all at once or "big bang."
- In adopting agility, it is recommended that organizations do NOT take this approach, but rather, adopt a more gradual change.
- Starting with a pilot Team to adopt an Agile method can be a great way for companies to realize where the impediments are and provide the opportunity to address them before rolling Agile out to another Team in the organization.

Assessment: Continuous Improvement

Questions:

1. True or False? Continuous improvement requires that we evaluate what is and is not working with the current process and then make adjustments we can immediately put into practice.
2. When an organization is adopting Agile, it is best to:
   A. Change everything all at once (big bang) ← Common
   B. Change gradually

TEKsystems
Our people make IT possible.

Impediments

Agile Anti-Patterns

- An "anti-pattern" is any behavior or practice that impedes your Agile adoption.
- These may seem useful at first glance, but in practice, they impede agility and your ability to achieve your Agile goals or deliver customer value quickly.
- These may also be referred to as "common problems" or "common reasons Agile fails."

- Instead of being open to something new, the mantra of "the way we've always done it" will start to be heard or the excuse of "the company's operational processes," "our change management process," etc.
- Staying entrenched in this type of thinking or practice can impede any Agile adoption and will set it up for failure—not success.

Impediments to Agile Success

1	Preserving Command and Control hierarchies over self-organization.
2	No active involvement from the business (Product Owner) or customer.
3	Documentation for documentation's sake and not for a useful goal or purpose.
4	Adopting Agile without defining goals or business problems that Agile will address, improve, or fix.
5	Unwillingness to invest in training and/or coaching.
6	Emphasis on metrics over project results.
7	Belief that Agile is for software development and not for business.
8	Over-forecasting in Iterations for fear the Development Team will not be busy enough.

- Several of the more common reasons why Agile adoptions or transformations fail are listed here.
- Discuss with other class participants: Do any of these resonate with you or your Team? Your company?

Determine Root Cause of Impediments

- Another common problem is turning work in Iterations or Sprints into "mini-waterfalls."
- This occurs when we stay locked in our roles, ignoring the fact that our goal is to deliver a working increment at the end of each Iteration or Sprint.
- What does it mean to be "code complete" at the end of a Sprint:
  - Does it mean the Backlog Items in the Sprint complete?
  - Does it imply that you are Test Complete?
  - Has your "customer" or Product Owner signed off and accepted the increment?
- If the Item is only code complete, Items are "carried over" for testing, acceptance, and refactoring into subsequent Sprints—thus, your mini-waterfall project!
- Basically, you are not finished with a feature until it meets the benchmark defined by done.

- In identifying root causes, Team members are encouraged and empowered to openly and honestly give visibility to what the problem is.
- Their Scrum Master, Coach, Agile champion, etc., should encourage these types of discussions.
- Those advocates can also be instrumental in facilitating discussions, so focus moves off of problem identification, or "ranting", about the issue and onto possible solutions and how to solve the problem(s).

Exercise: Recognize Anti-Patterns

Instructions:

- Self-organize into Teams, even if you are with different Team members from previous exercises.
- Your instructor will provide scenarios from Agile projects.
- Your Team's job is to discuss and identify the anti-patterns and what might be the root cause of the problem.

Anti-Pattern Scenario #1

- At the end of each Sprint, there is always a pile of Stories "not complete" or that did not reach the "Definition of Done."
- Instead of putting incomplete items back on the Product Backlog, the Development Team insists on getting "credit" for what WAS done and splits the Stories.
- This artificially inflates the Development Team velocity and keeps the Development Team in this cycle of carrying over Technical Debt from Sprint to Sprint.
- Is this optimal? Why or why not?
- What anti-patterns do you see?
- What would you do differently?

Anti-Pattern Scenario #2

- Your Development Team has decided to use Story Points and uses cards to collaboratively reach consensus on estimates.
- The Product Owner and manager attending planning sessions routinely interject, and have asked for a points-to-hours "conversion" so they can more effectively manage the Team's work.
- Is this optimal? Why or why not?
- What anti-patterns do you see?
- What would you do differently?

Anti-Pattern Scenario #3

- Your Development Team has been together for six 2-week Sprints to date with a running average velocity of 76 Story Points.
- With the holidays coming up, the Scrum Master sends out new meeting invitations extending one Sprint for several days and shortening another Sprint to "get around" holidays.
- Is this optimal? Why or why not?
- What anti-patterns do you see?
- What would you do differently?

Anti-Pattern Scenario #4

- The Tech Lead on your Team is also the Scrum Master. Scrums run over, planning sessions run over due to lack of preparation, impediments have not been addressed for some time, the Product Owner is rarely ready for refinement sessions, etc.
- Coding tasks given to a "Tech Lead/SM" are sometimes late, and other times are contributing to Technical Debt for the Dev Team.
- Is this optimal? Why or why not?
- What anti-patterns do you see?
- What would you do differently?

Anti-Pattern Scenario #5

- Your Development Team has been together for a little over a year adopting Scrum and blending in some XP practices.
- The group has decided Sprint Retrospectives should be cancelled since there are rarely impediments or issues any more.
- Is this optimal? Why or why not?
- What anti-patterns do you see?
- What would you do differently?

Anti-Pattern Scenario #6

- During Sprint Planning, the Development Team has gotten into the habit of planning "the entire Sprint," assigning specific tasks to specific people based on comfort zone, history, etc.
- At the end of the Sprint, there are several Stories that are not done based on the "silos" created by individuals working on the tasks for those Stories.
- Is this an optimal scenario? Why or why not?
- What anti-patterns do you see?
- What would you do differently?

Discussion: Anti-Patterns

Questions:

- Based on our exercise around anti-patterns and impediments, what challenges do you see in your real-life Team adopting Agile? Your organization?
- What are some possible choices your Team or your organization can make to correct these?

Assessment: Impediments

Questions:

1. True or False? A company can successfully implement Agile without any active involvement from the business or customer.
2. True or False? A common problem in Agile is turning work in Iterations or Sprints into mini waterfalls.

What Now?

Shu Ha Ri

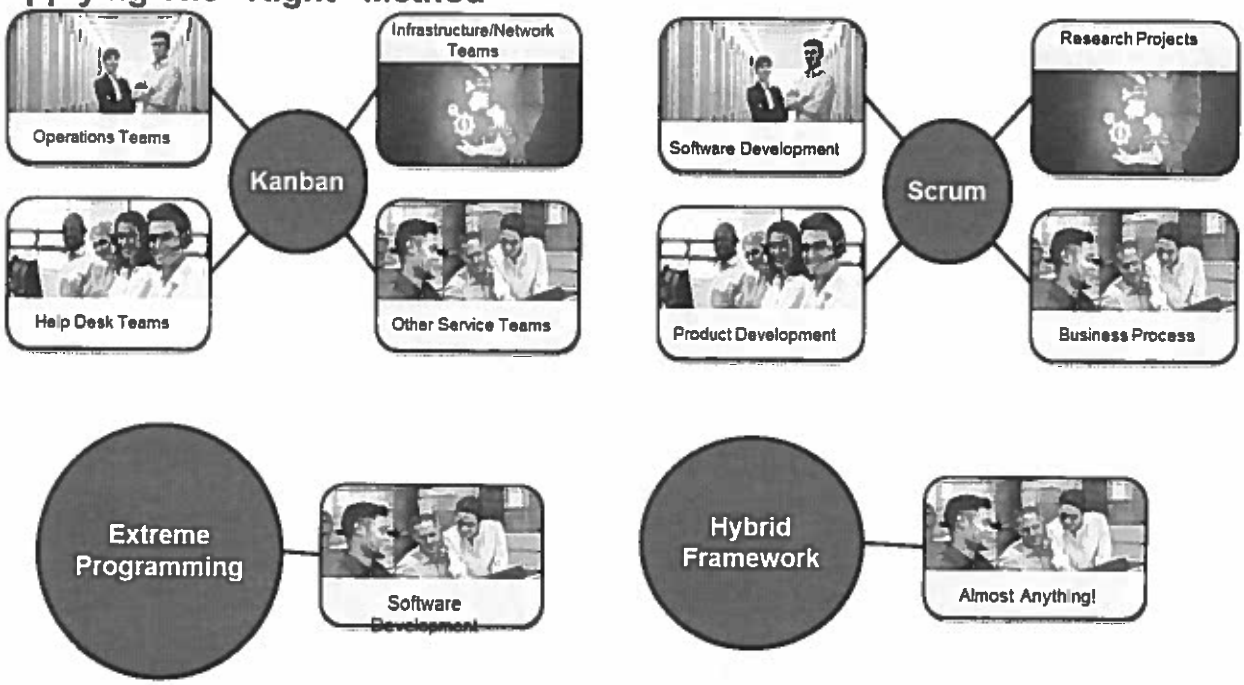
Learning, applying, and customizing new practices can be a challenge:

- "**Shu Ha Ri**" is a learning technique that comes from the Japanese martial of Aikido:
  - **Shu** "*learn*": The student has a "master" that provides guidance. The student follows the master—not worrying about alternatives, underlying theory or nuances.
  - **Ha** "*detach*": The student is now able to apply the basic practices and start to learn underlying principles and theories. Other masters are identified and some of their teachings are incorporated.
  - **Ri** "*transcend*": The student now learns from her own practice and efforts. She creates new approaches and adapts practices to her situation.
- Shu Ha Ri can be applied in your own growth in learning about Agile.
- Shu Ha Ri can be used as you learn, adopt, and succeed with Agile.

- Further discussion of Shu Ha Ri can be found via:
  - http://agilecoach.typepad.com/agile-coaching/2010/02/shuhari-considered-harmful.html.
  - http://c2.com/cgi/wiki?ShuHaRi.
  - http://alistair.cockburn.us/Shu+Ha+Ri.
  - http://martinfowler.com/bliki/ShuHaRi.html.
- Remember that when working with the Master, they are not dictating a one-size-fits all approach to the adoption of the new skills. They are aware of the context and are guiding you to useful skills and practices. In the early stages you are placing trust in the master to guide you to appropriate practices. Later, as you gain experience and mastery, you start to develop new approaches.

守破離

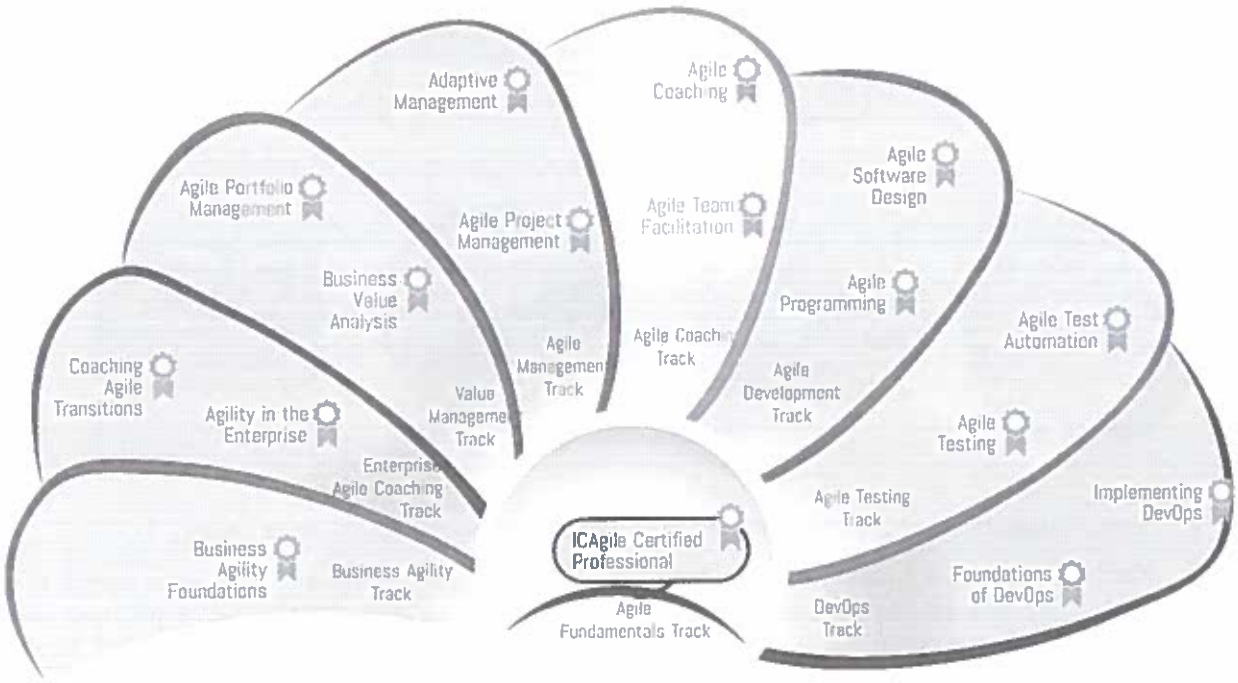
Applying The "Right" Method



- Many Teams find that a particular Agile method may be more suited to their needs
- **Kanban:** Kanban is typically more suited to service teams, help desks and operations teams. Any process serving a continuous stream of relatively independent tasks may be well suited to Kanban.
- **Scrum:** Scrum is typically more suited to product development or research based projects. Any process serving groups of co-dependent task that all add up to a lager deliverable product my be well suited to Scrum.
- **Extreme Programming:** Extreme programming can be very effective in a software development project. It is rare to see this Agile method employed outside the realm of software development.
- **Hybrid Methods:** Many Teams find it best to use a hybrid method. These Teams borrow practices from several Agile methods to create a custom method specifically suited to their needs. One example, referred to as "ScrumBan," uses the Kanban framework and borrows practices from Scrum such as the daily standup meeting, retrospectives and the roles of Product owner and Scrum Master. Teams using this method may also borrow practices from Extreme programming such as Test Driven Development and Pair Programming.

Agile Beyond Software Development

ICAgile Roadmap highlights additional areas of Agile: Leadership, Coaching, Portfolio Management, Project Management, Facilitation, Business Agility



In this course, we're prepared you for receiving the ICAgile Certified Professional designation. This is the foundation for all of the other ICAgile certifications. Take a moment to review the diagram and note that many other topics are covered in the program. This highlights that there is more to Agile than just Software Development.

Discussion: Agile Beyond Software

Question: Suppose you are putting together a Team of writers to create a monthly blog website. Which Agile practices would you adopt? Would you have to modify them? If so, how?

- ❑ The 12 Agile Principles
- ❑ Transparency, Inspection, Adaptation
- ❑ Stand Up Meeting
- ❑ Retrospective
- ❑ Acceptance Tests
- ❑ User Stories
- ❑ Demo / Sprint Review
- ❑ Test-Driven Development (TDD)
- ❑ Other?

Agile values and practices have been used outside software products and projects. Such projects range from sales and marketing campaigns to running businesses to educating students. As you think about non-software projects, you may feel some of the items on this slide do not apply. But it may surprise you how many do apply, or can with a little adaptation. The blog article "Agile marketing helps you innovate faster – but it's not right for every team" provides some insight on marketing teams adapting Agile for their own needs. https://www.teksystems.com/en/resources/teksavvy-blog/2016/agile-marketing-not-right-for-everyone

Exercise: Next Steps

Instructions:

- Discuss your next steps or plan of action for improvement in your job or role.
- Use whatever materials you need to facilitate this discussion: index cards, sticky notes, flip chart paper, etc.
- This process should include identification of the next steps, any order or priority for tackling them, and timelines if at all possible.
- Discuss with your instructor as needed.

Discussion: Next Steps

Questions:

- What did you come up with?
- Could you use more time to complete this exercise? If yes, you may want to schedule time outside the class to do this, especially if there are other roles, such as a Product Owner, needed for the discussion.

Recap

Let's Recap

- What Agile values and principles are most important to you and your Team?
- Based on the methods we've discussed, which Agile method do you think will work best for your Team and organization? Why?
- What are your Goals in Adopting Agile?
- Who is your customer?
- What is your Product, Service, or Project Vision?
- What is the difference between a Product Backlog and a Sprint Backlog?
- What Estimation method is your Team or organization most likely to use, and why?
- What levels of planning are standard for Agile?
- Why is it important to continuously review and adapt at all levels of the Agile process?
- How can your organization change incrementally to ensure the success of Agile?
- What impediments to success do you think you may need to overcome in your organization?

Appendix A: Continuing Education

Claiming Scrum Alliance SEUs

Claiming Scrum Education Units (SEUs):

- If you already have a Certified Scrum Master®, Certified Scrum Product Owner®, or Certified Scrum Developer® certification, the hours in this class will count towards your continuing education needs to advance to or maintain the next level: Certified Scrum Professional® (CSP).
- Check the Scrum Alliance website at http://www.scrumalliance.org for the latest requirements and to claim your SEUs.

Certified Scrum Master®, Certified Scrum Product Owner®, Certified Scrum Developer®, and Certified Scrum Professional® are registered trademarks of the Scrum Alliance

Claiming PMI PDUs

Claiming PMI Professional Development Units (PDUs):

- As a Project Management Professional (PMP) or PMI-Agile Certified Practitioner (ACP), you may be eligible for Professional Development Units (PDUs) or qualifying educational hours from taking this course.
- You may claim 1 PDU for each 1 contact hour of this course that is considered an applicable activity for the respective credential that you hold.
- For more information about claiming PMI PDUs, refer to the document "Steps to Report PDUs" provided with your training materials for this course.

Thank you for your time!

Let's clear our Product Backlog!

TEKsystems

Our people make IT possible.

TEKsystems Education Services would like to thank you for attending this class. We hope it has met your expectations.

Please go to www.metricsthatmatter.com/teksystems to complete the final course evaluation. We greatly appreciate your feedback.

Thank you!

Sincerely,

TEKsystems Education Services Team

Learn more at **TEKsystems.com**

TEKsystems
Education Services