TEKsystems

Our people make IT possible.

TEKsystems Learning Solutions

presents

Agile Team Training Camp - Agile Certified Professional

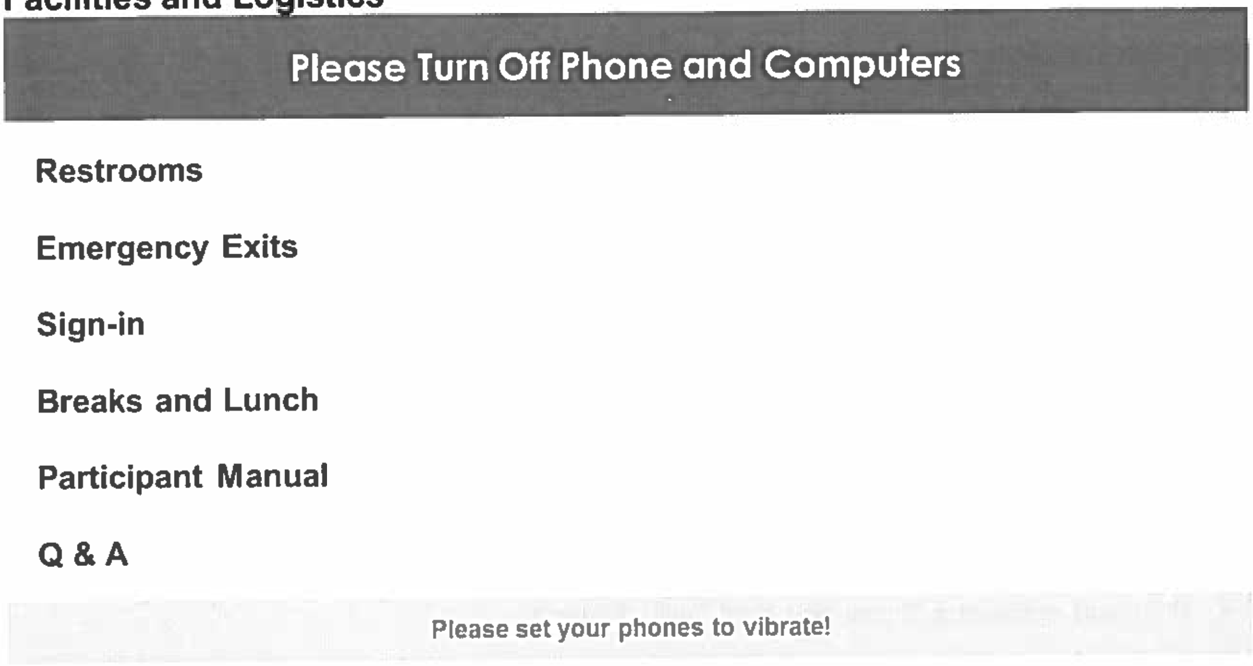
Participant Guide

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

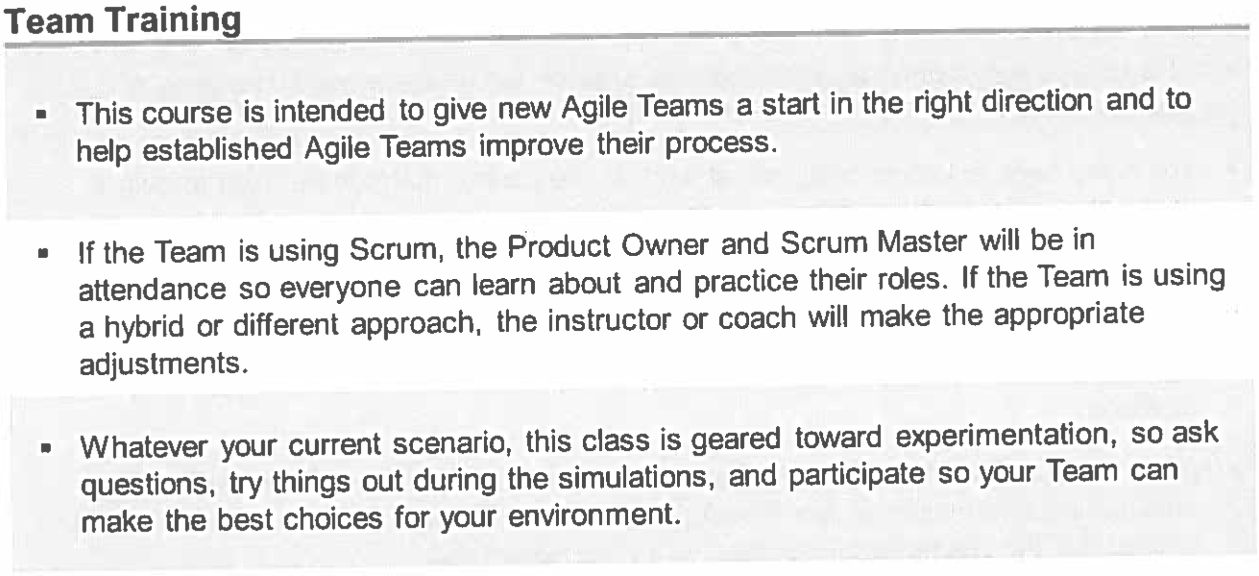
Please set your phones to vibrate!

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.

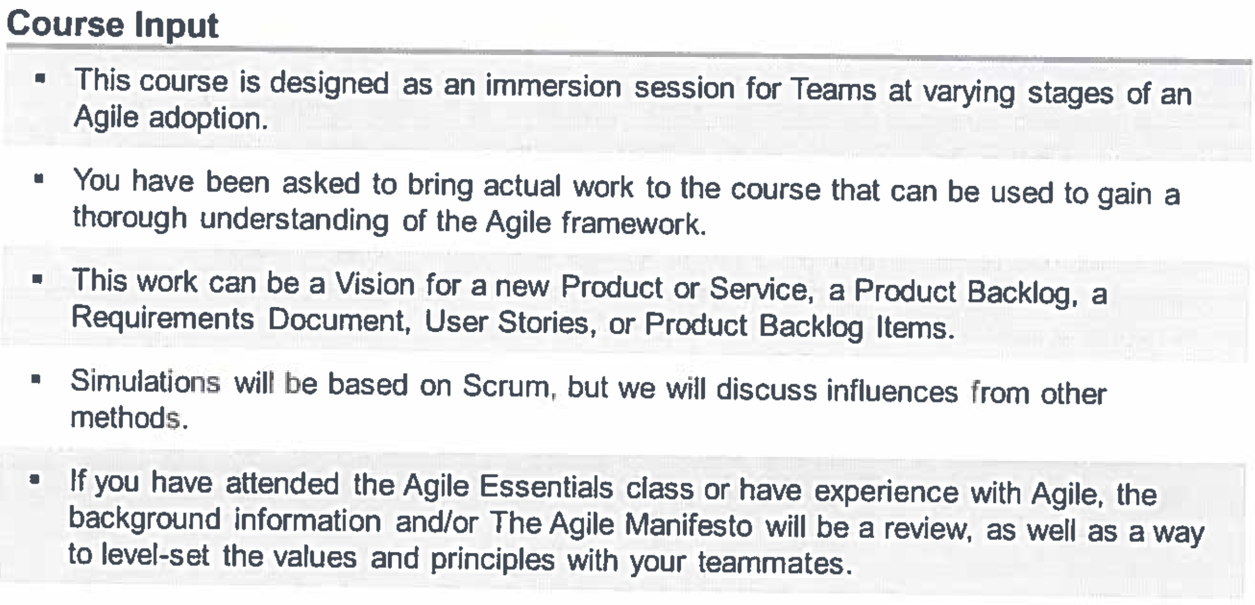
- In addition to the objectives you identified in taking this course, these are the ones we expect to achieve based on the material provided.



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.

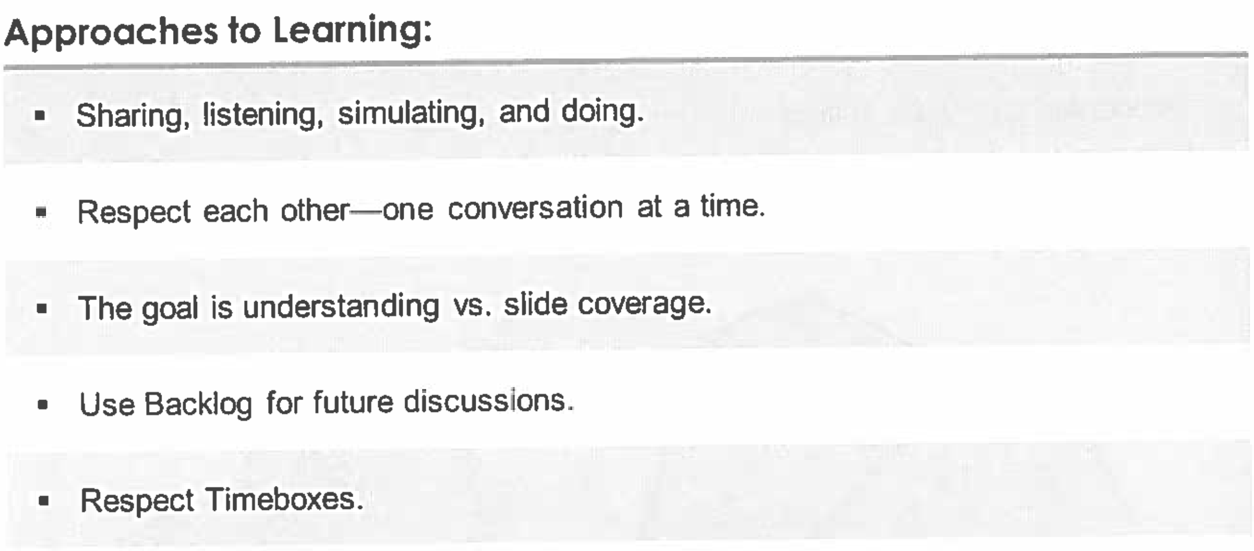


Course Input

- 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:

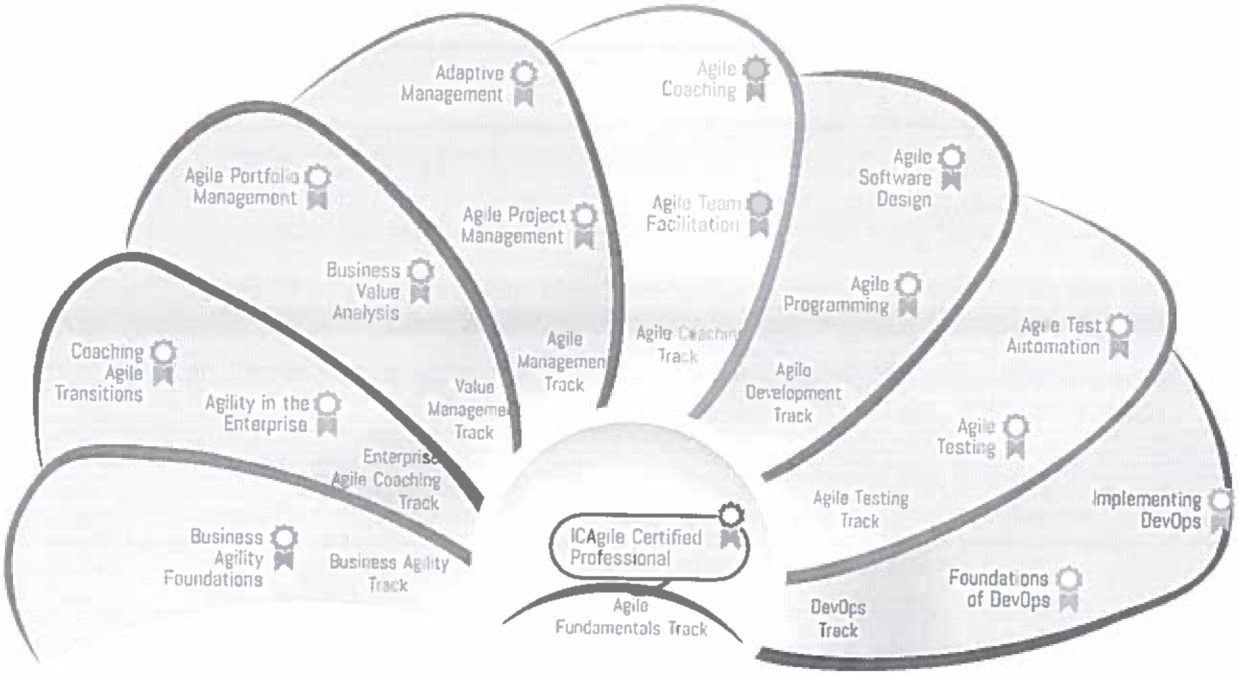
- Sharing, listening, simulating, and doing.
- Respect each other—one conversation at a time.
- The goal is understanding vs. slide coverage.
- Use Backlog for future discussions.
- Respect Timeboxes.

- 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:

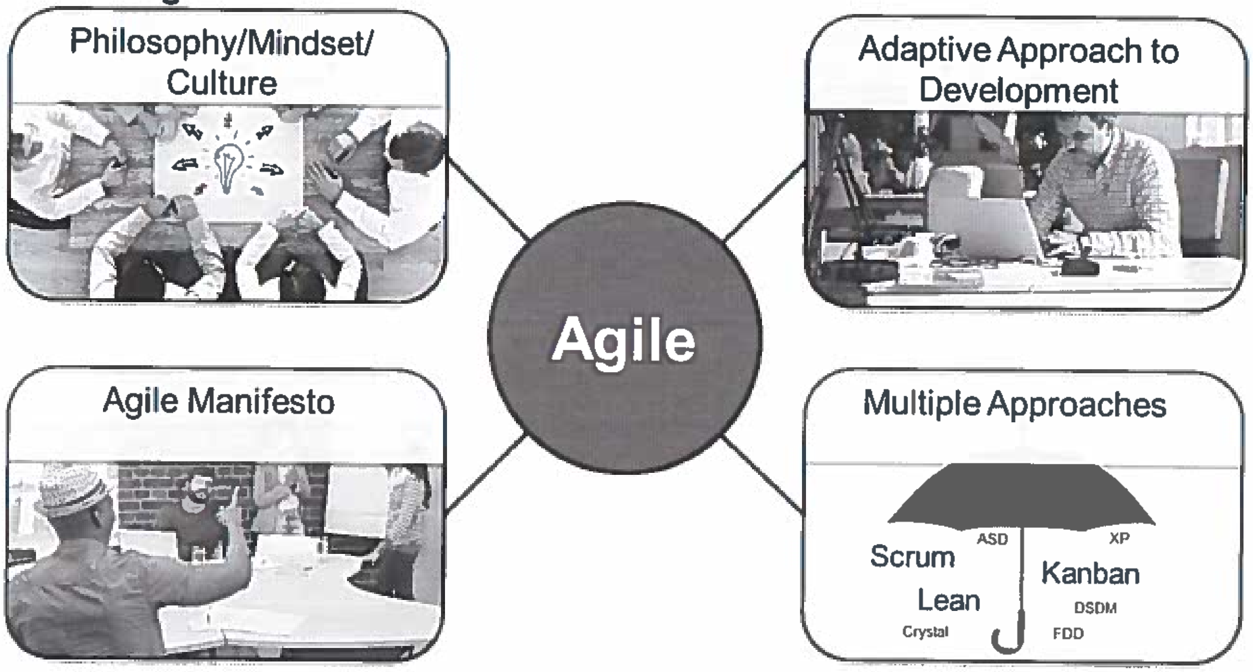
- Attend and participate in class.
- Complete module assessments.



- 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?



Philosophy/Mindset/ Culture

Adaptive Approach to Development

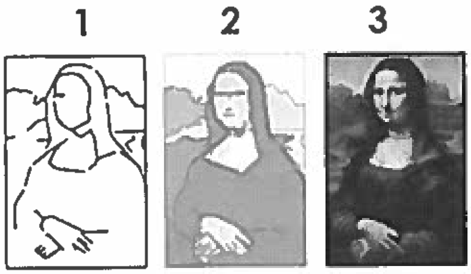
Agile

Agile Manifesto

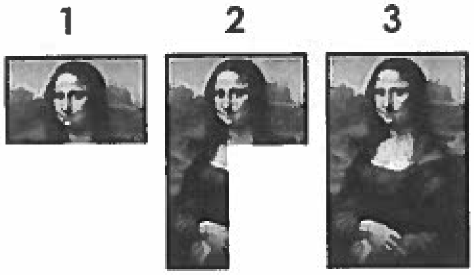
Multiple Approaches

- 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

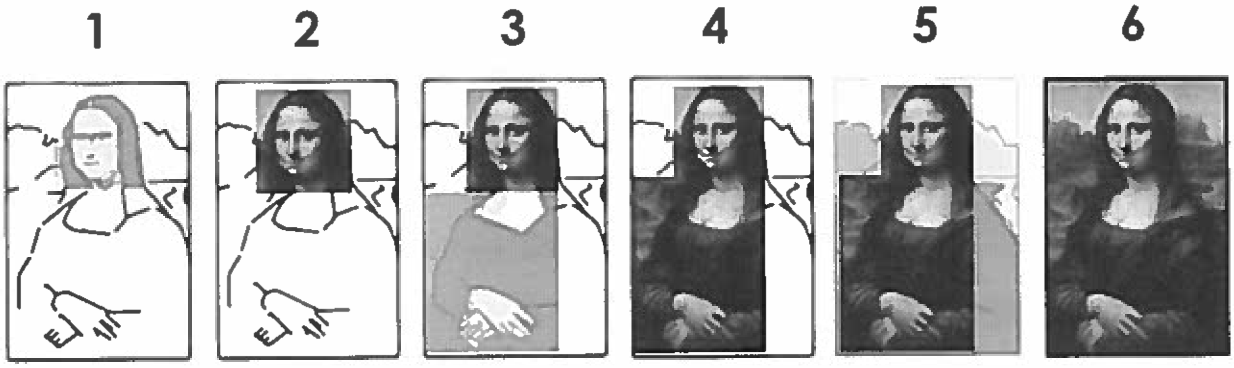


Iterative



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

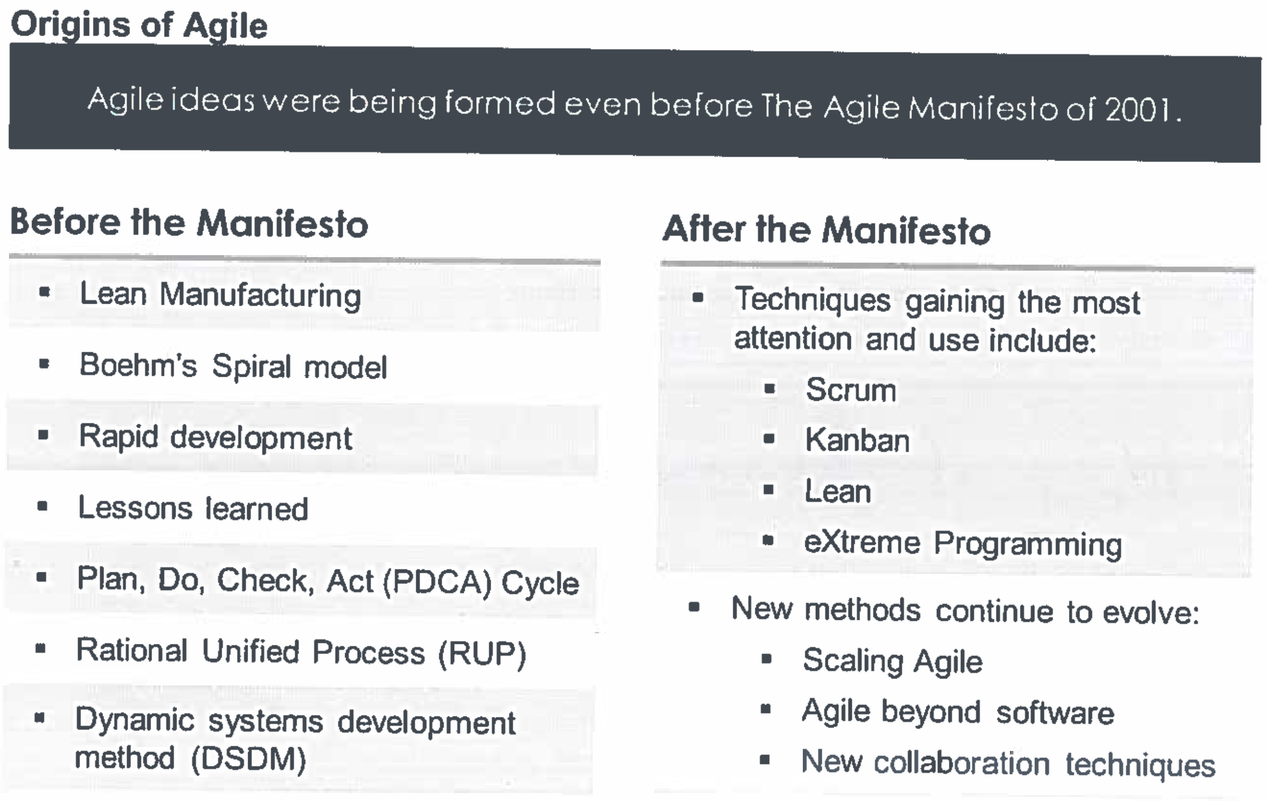
10 min

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?

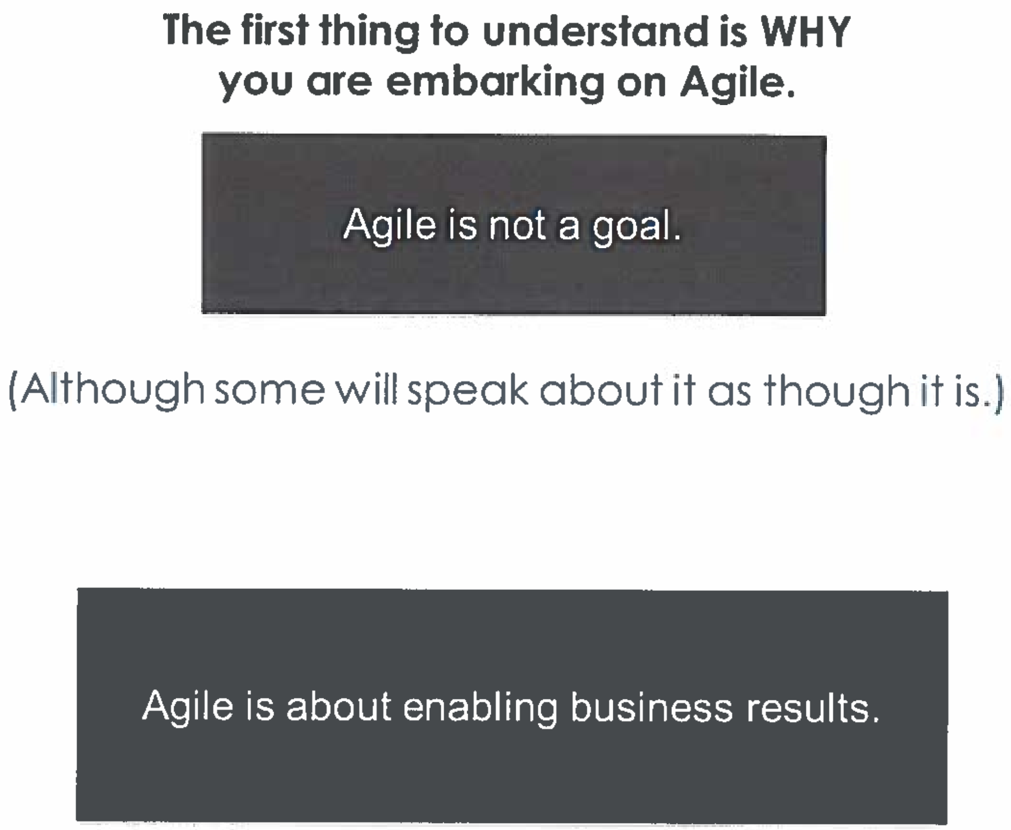
Questions:

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?

Section 2

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

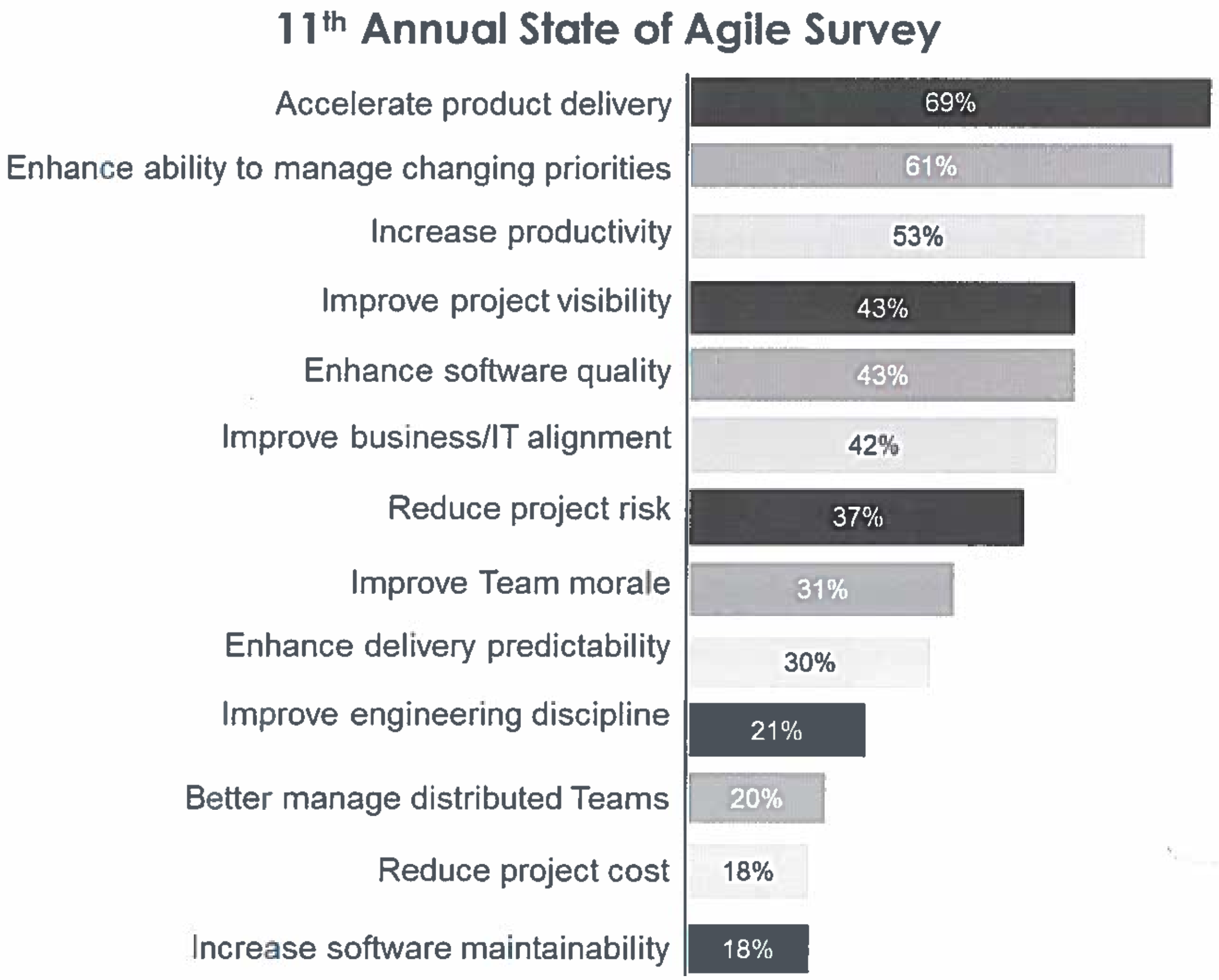
10 min

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

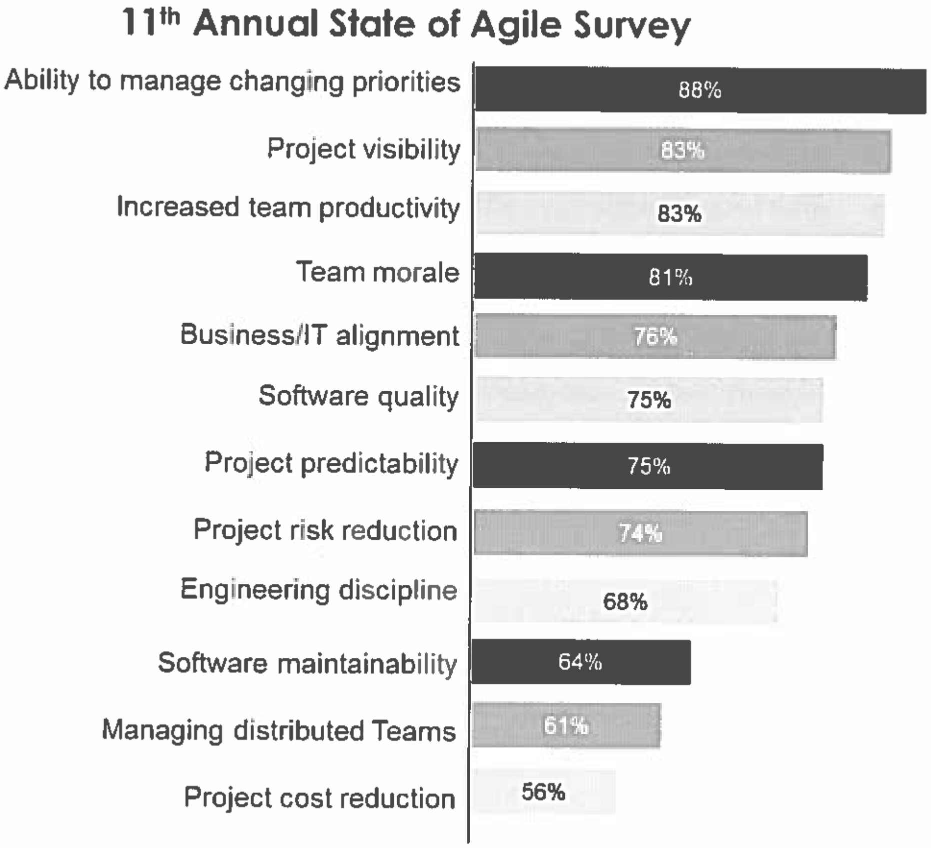


Source: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 (page 8)

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

10 min

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:

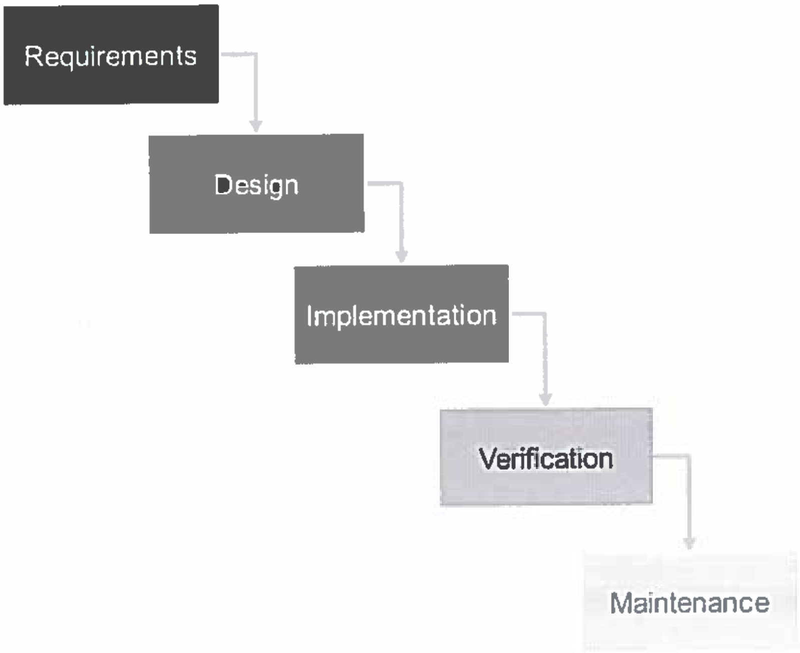
1. True or False? Agile is about enabling business results.
2. True or False? Agile will quickly expose impediments and problems the project or organization already has.

How is Agile Different?

Section 3

Traditional Process: Waterfall

> "I believe in this concept but the implementation is risky and invites failure…"
>
> – Dr. Winston Royce



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?

5 min

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.

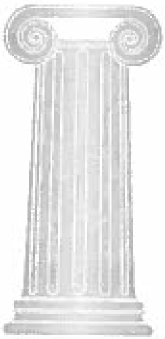
Agile: an Empirical Process

An empirical process embraces change as opposed to discouraging it.

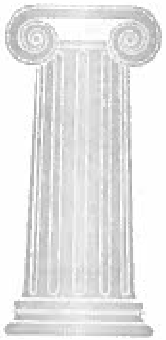
This means that we do not necessarily know a lot of details up front, but rather we learn as we go.

The 3 pillars of any empirical process are:

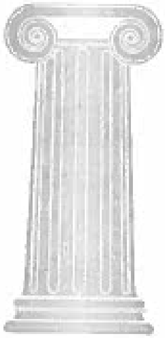
Transparency



Inspection



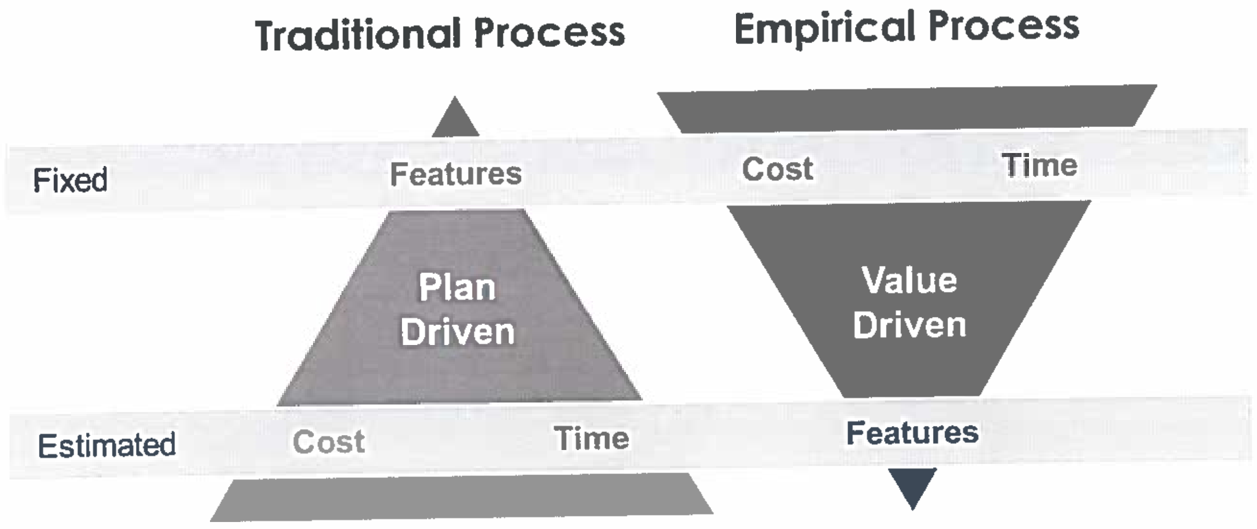
Adaptation



- 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.

Section 3

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?

5 min

Transparency
Inspection
Adaptation

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.

Agile Approaches

Section 4

Agile Methods

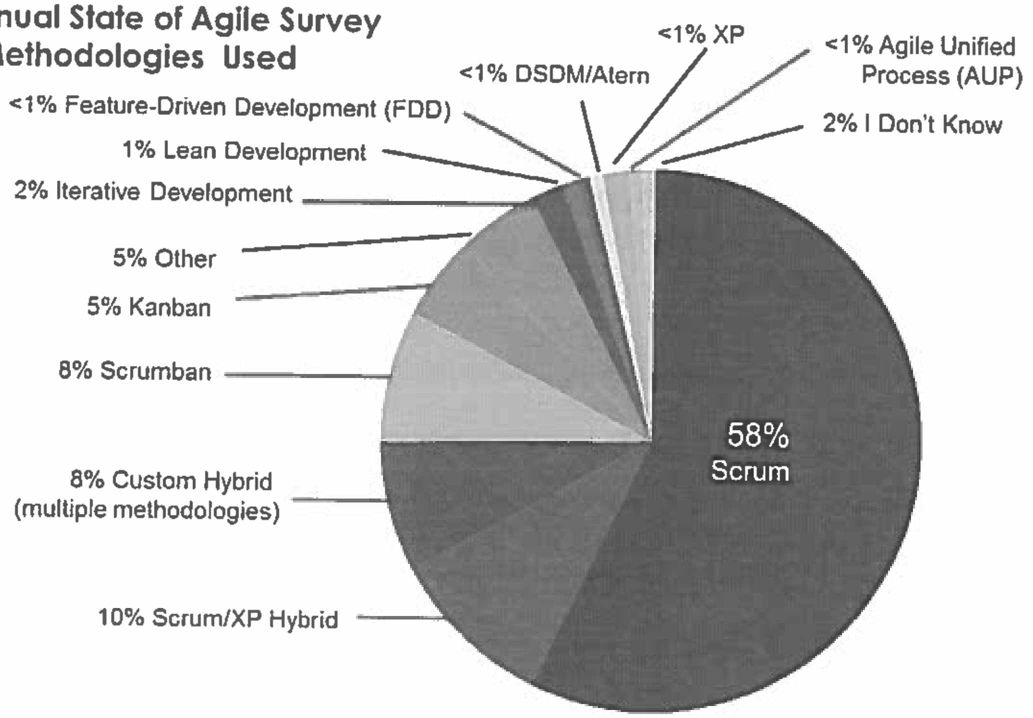
Many frameworks and approaches fall under the "Agile umbrella."

- Lean Software Development
- Kanban
- eXtreme Programming
- Scrum

- 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

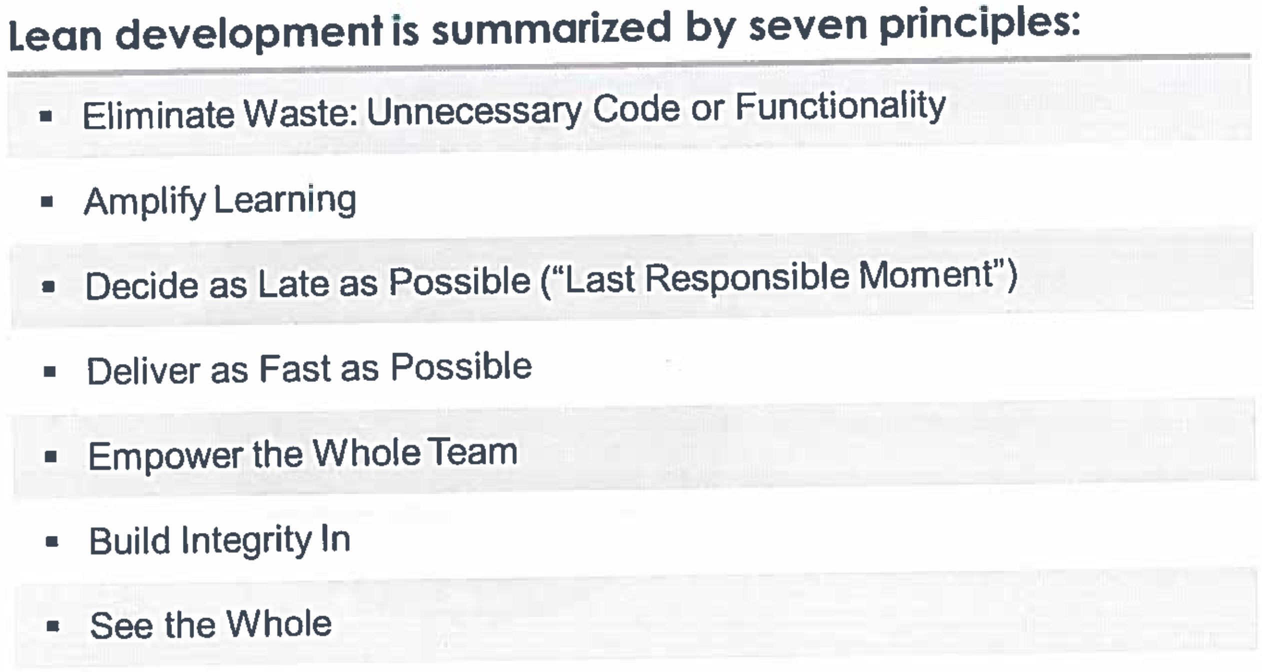


Source: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 (page 10)

- 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:

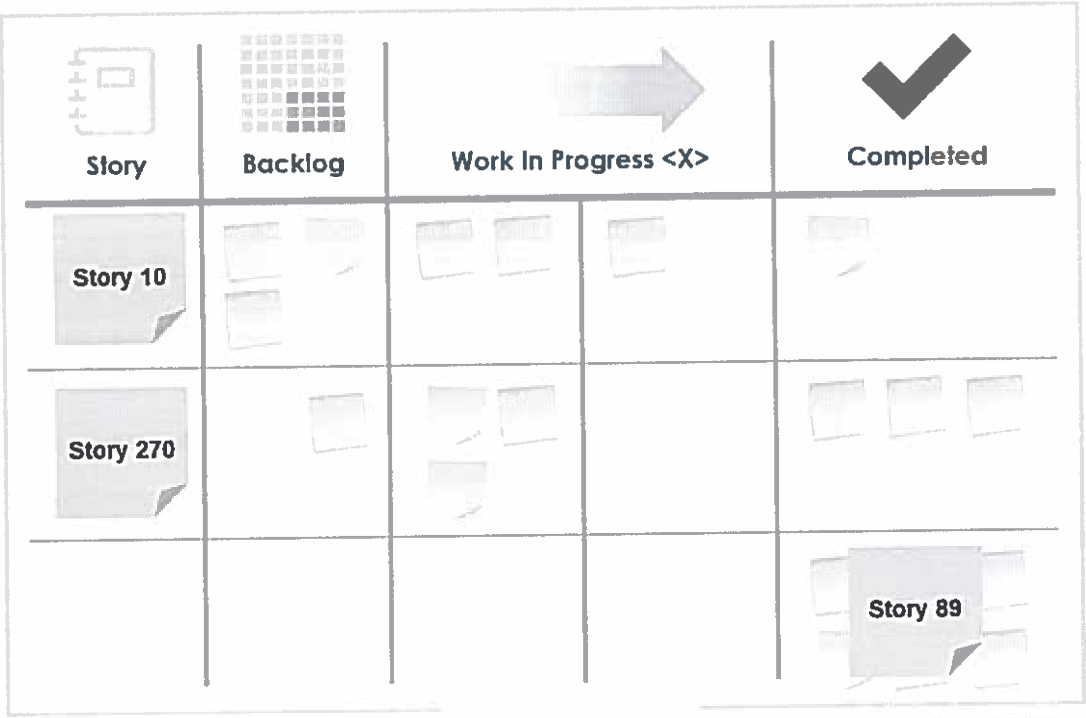
- Eliminate Waste: Unnecessary Code or Functionality
- Amplify Learning
- Decide as Late as Possible ("Last Responsible Moment")
- Deliver as Fast as Possible
- Empower the Whole Team
- Build Integrity In
- See the Whole

- 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

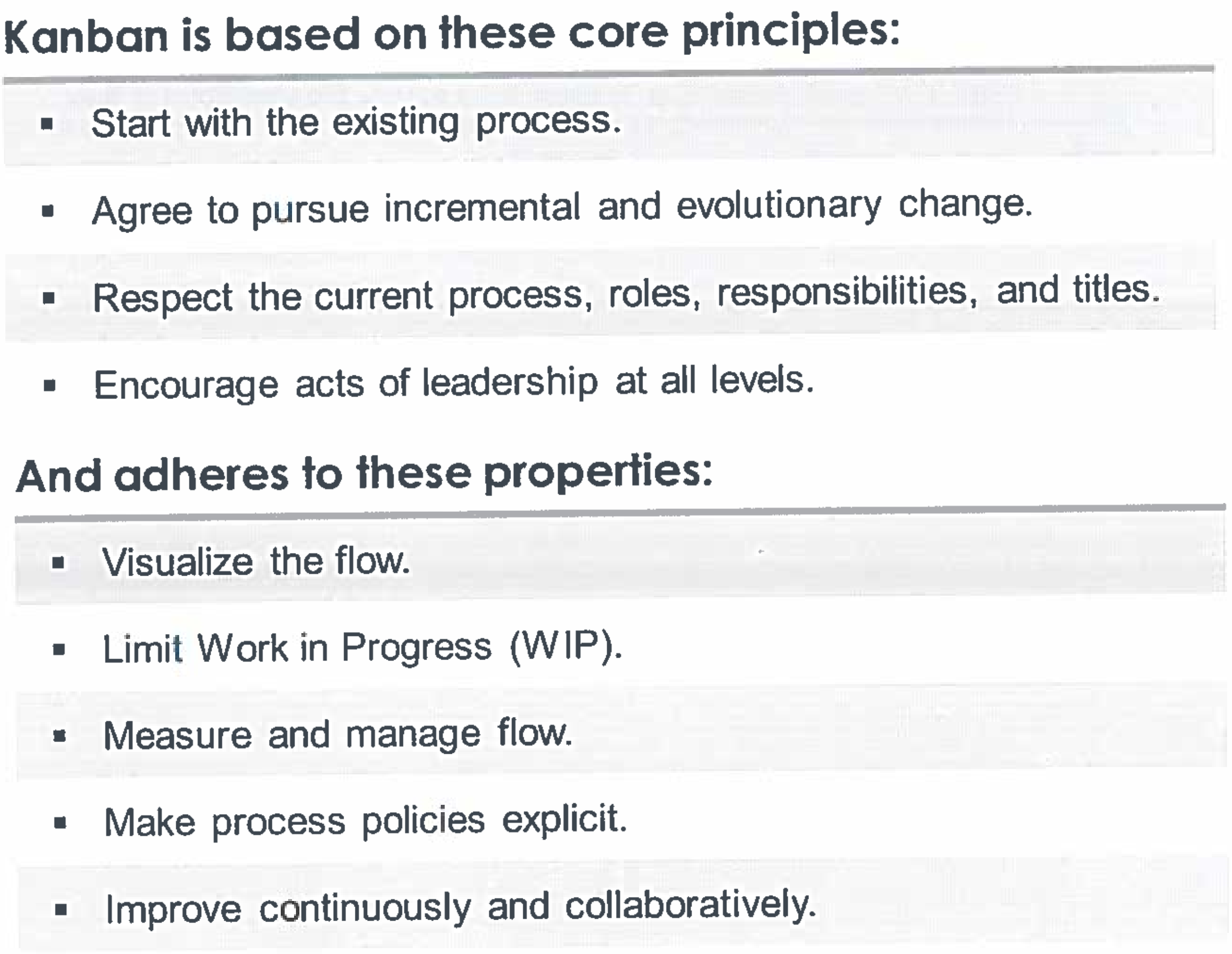


Explicit Policies and Exit Criteria:	**Definition of Ready**	**I.N.V.E.S.T.**	**WIP limit = 6. Acceptance Criteria met and UAT completed. Definition of Done.**	**Items remain visible for 14 days then get archived**

- 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.

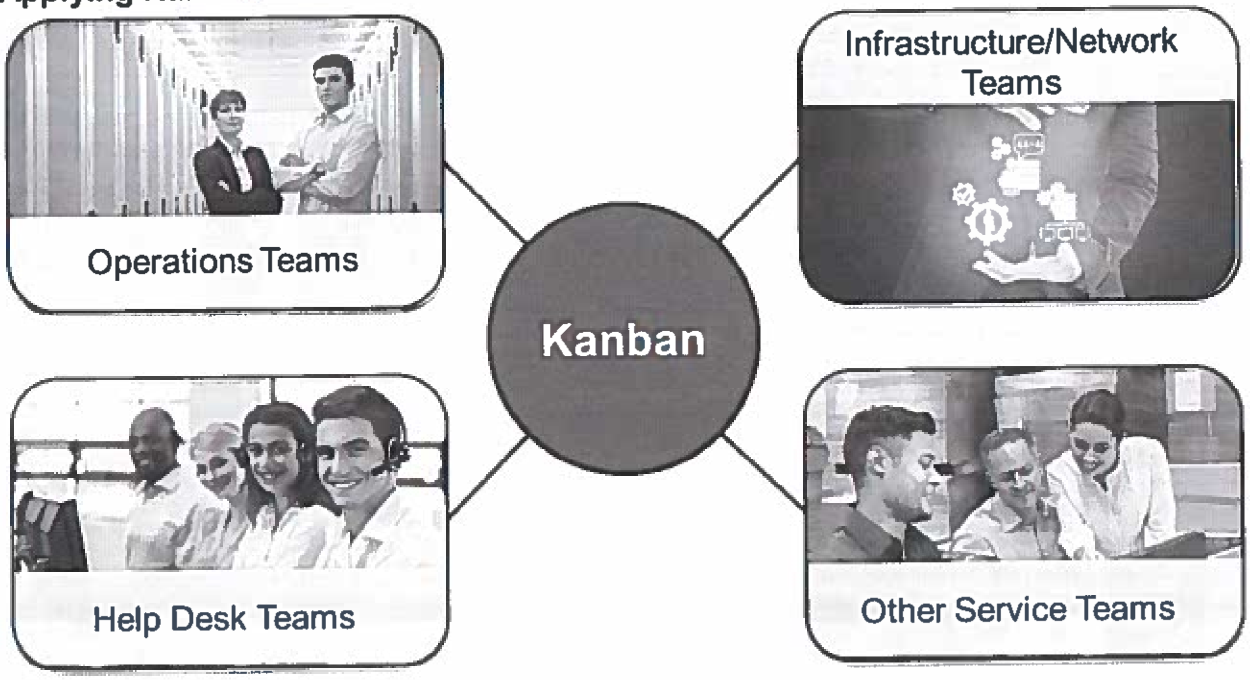
Kanban-Successful Evolutionary Change for your Technology Business, by David Anderson

- 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

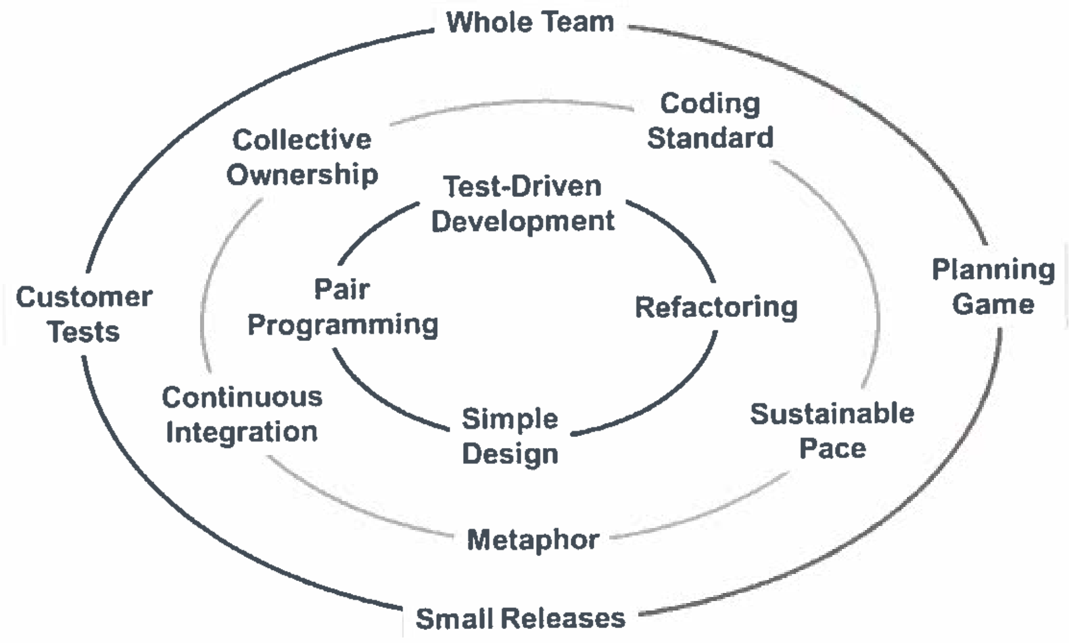
15 min

Questions:

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

Section 4

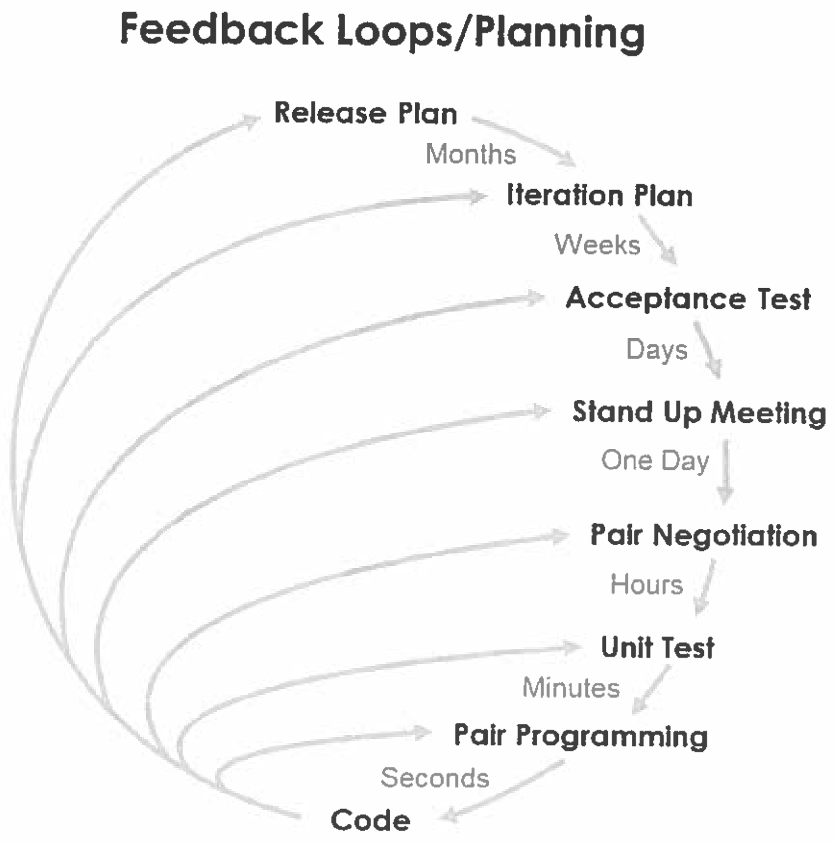
eXtreme Programming (XP)



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

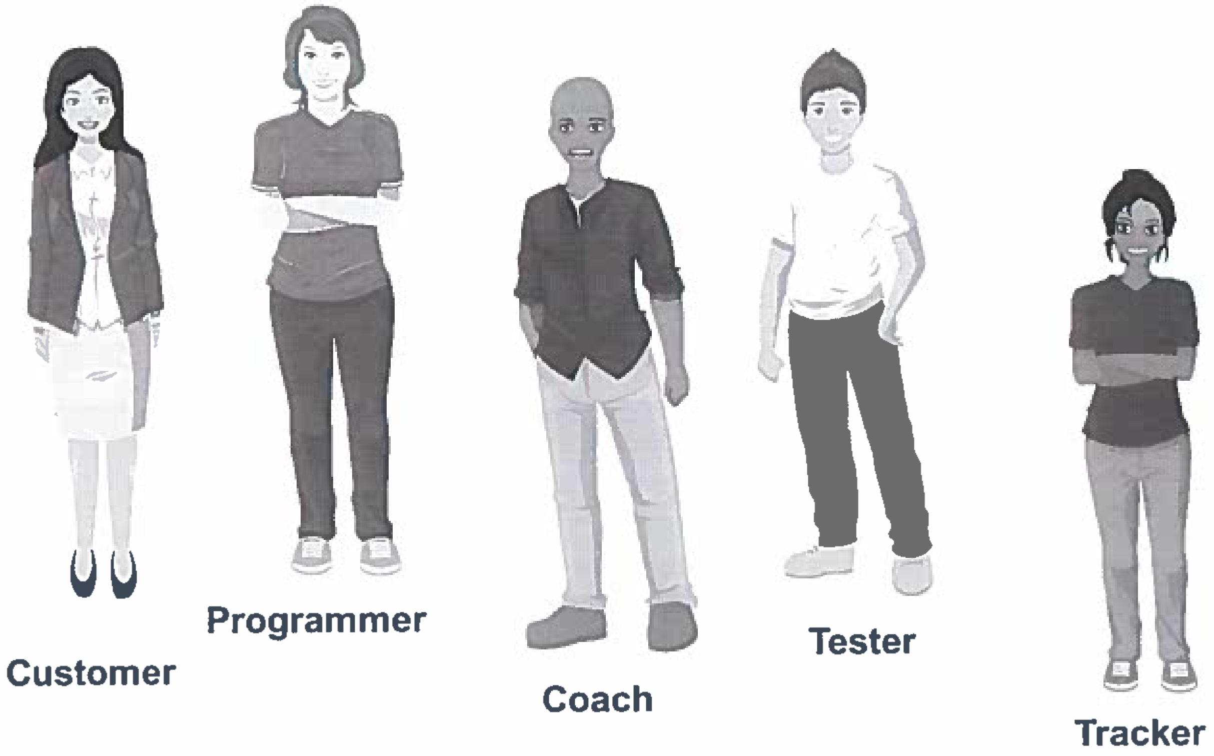
- eXtreme Programming (XP) is another method under the Agile umbrella. It was created by Kent Beck during his work on the Chrysler Comprehensive Compensation System (C3) payroll project. He became the C3 project leader in March 1996 and began to refine the development method used in the project. He wrote 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.

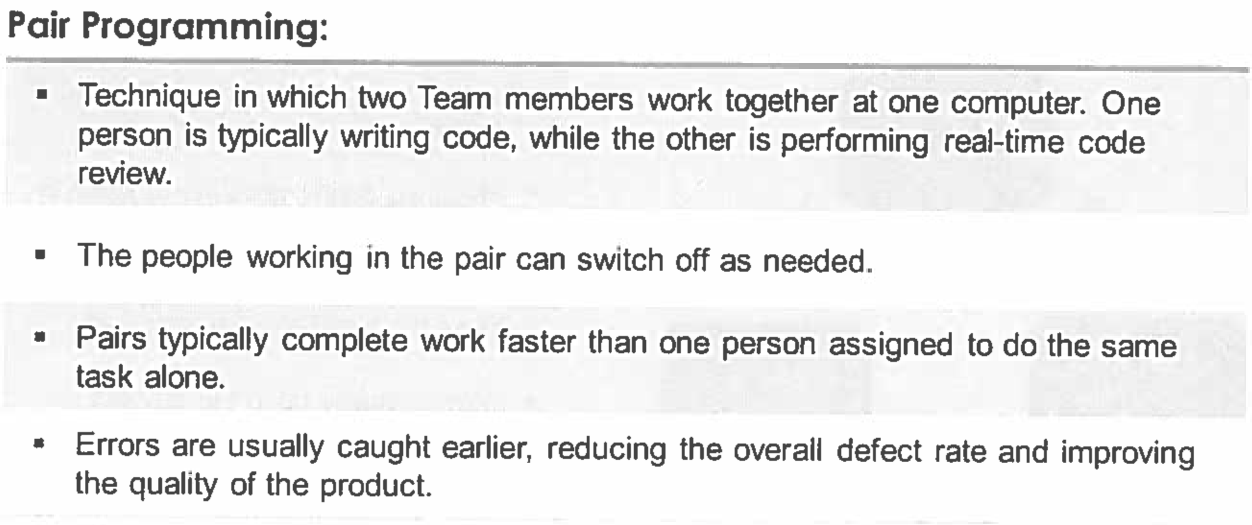
XP Roles



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



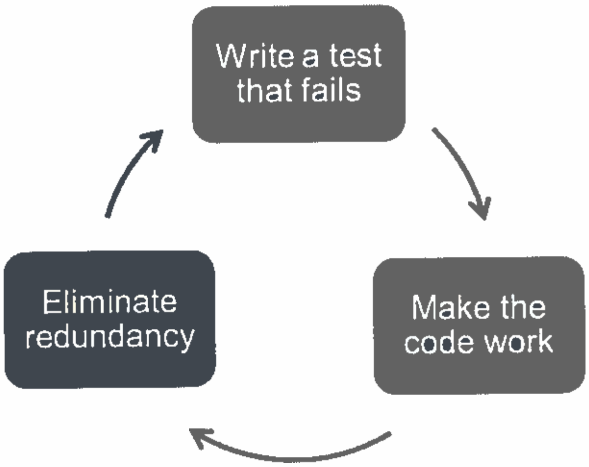
Pair Programming:

- 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?

Section 4

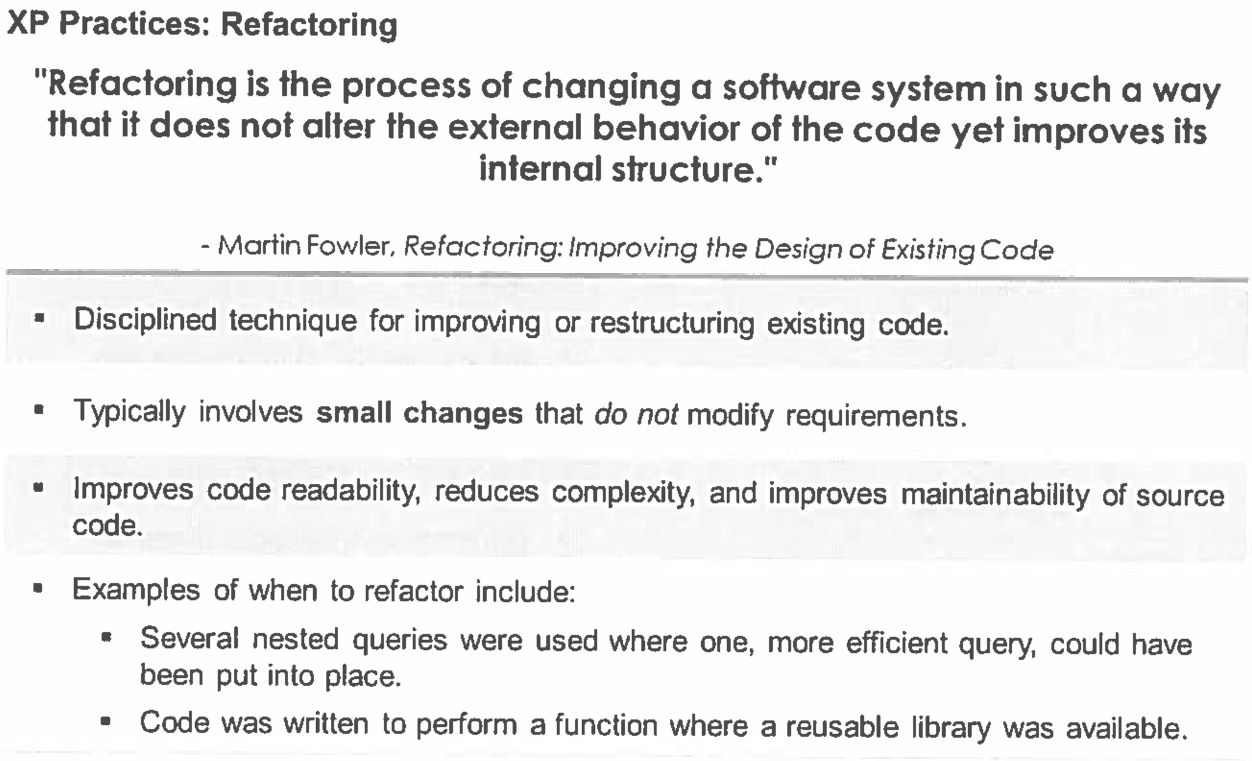
XP Practices: TDD



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.



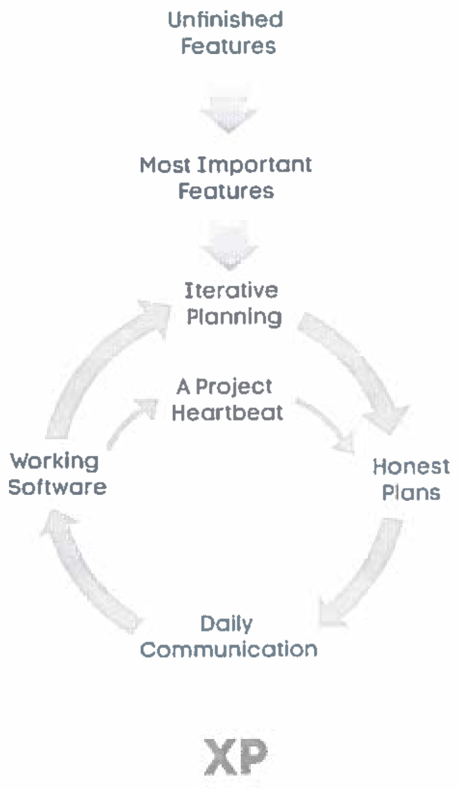
XP Practices: Refactoring

> **"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.

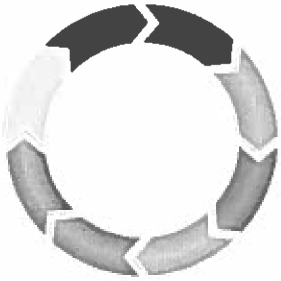
Source: http://www.extremeprogramming.org

- 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)

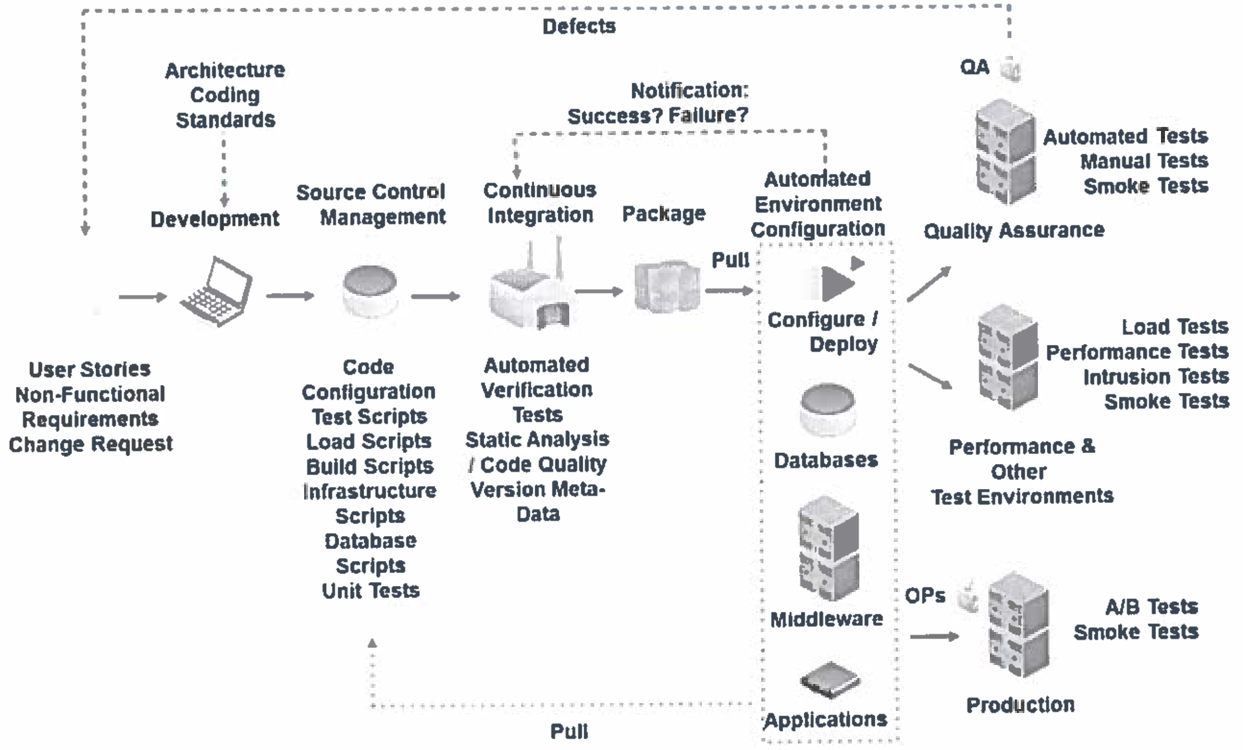
Taking CI one step further:

- **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

10 min

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?

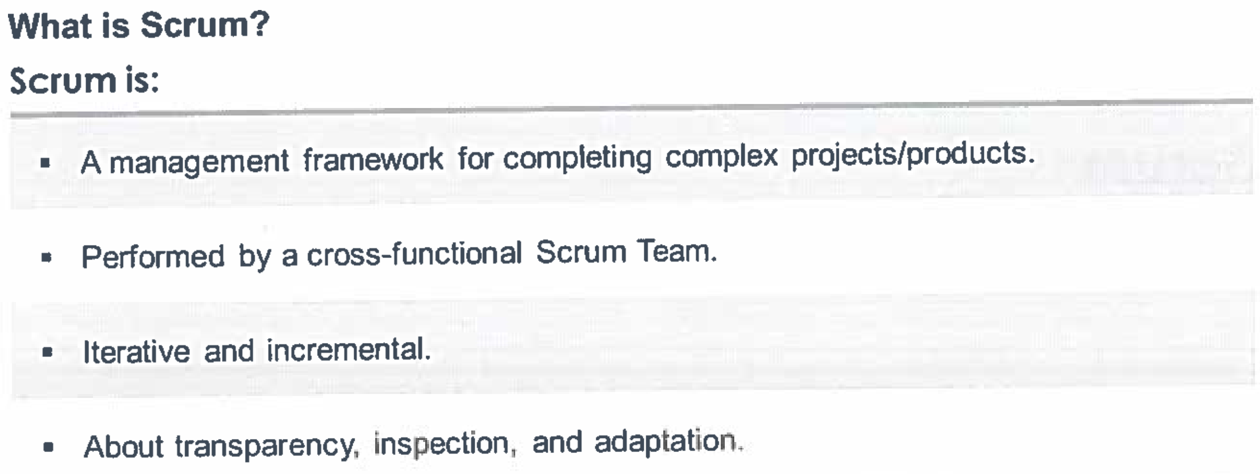
   A. Pair Programming

   B. TDD

   C. Refactoring

   [✓ D.] Continuous Integration

Scrum



What is Scrum?

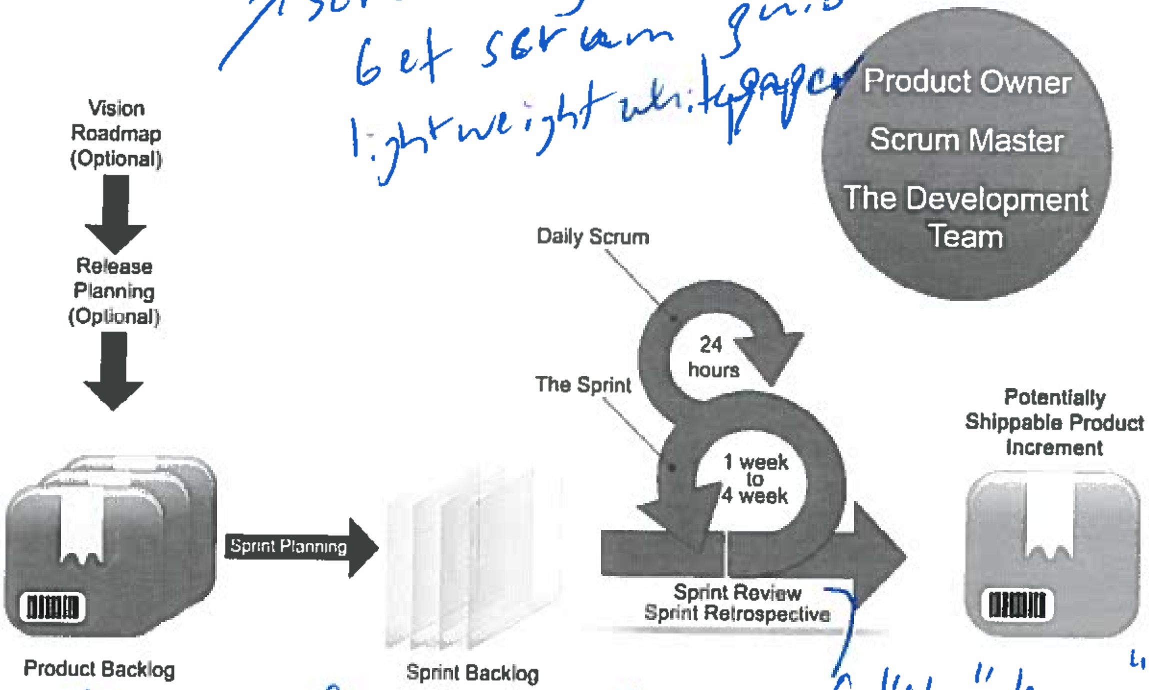
Scrum is:

- 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

↗ scrum.org
Get scrum guide
lightweight whitepaper



↑ wish list for product

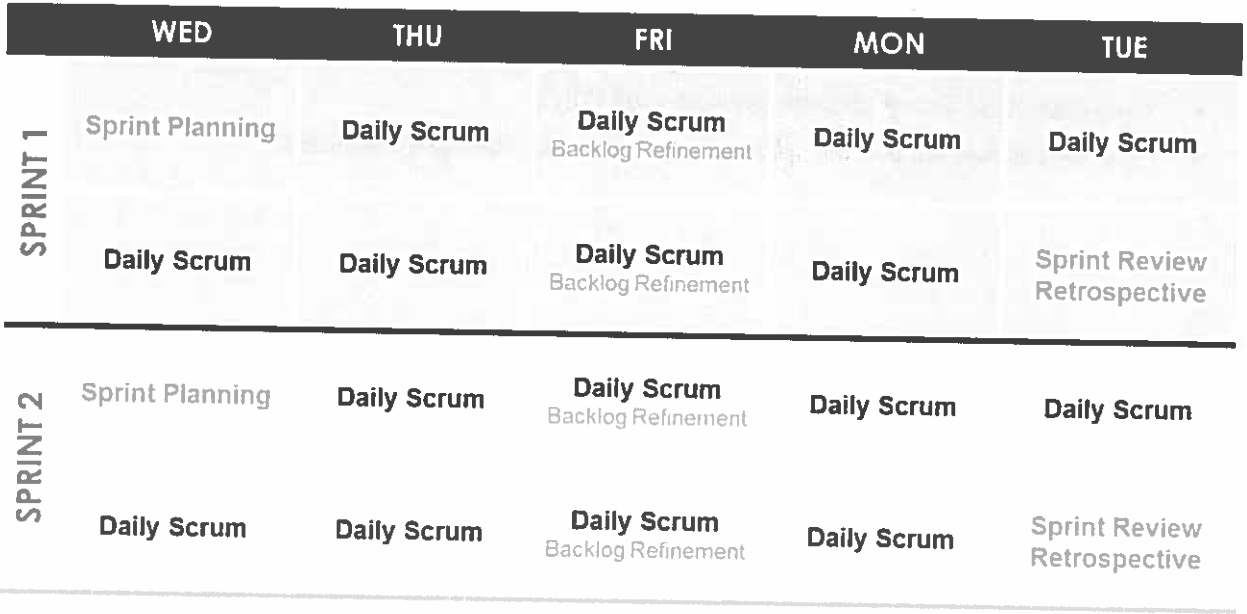
called "demo" for stakeholder review what you did for sprint

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?

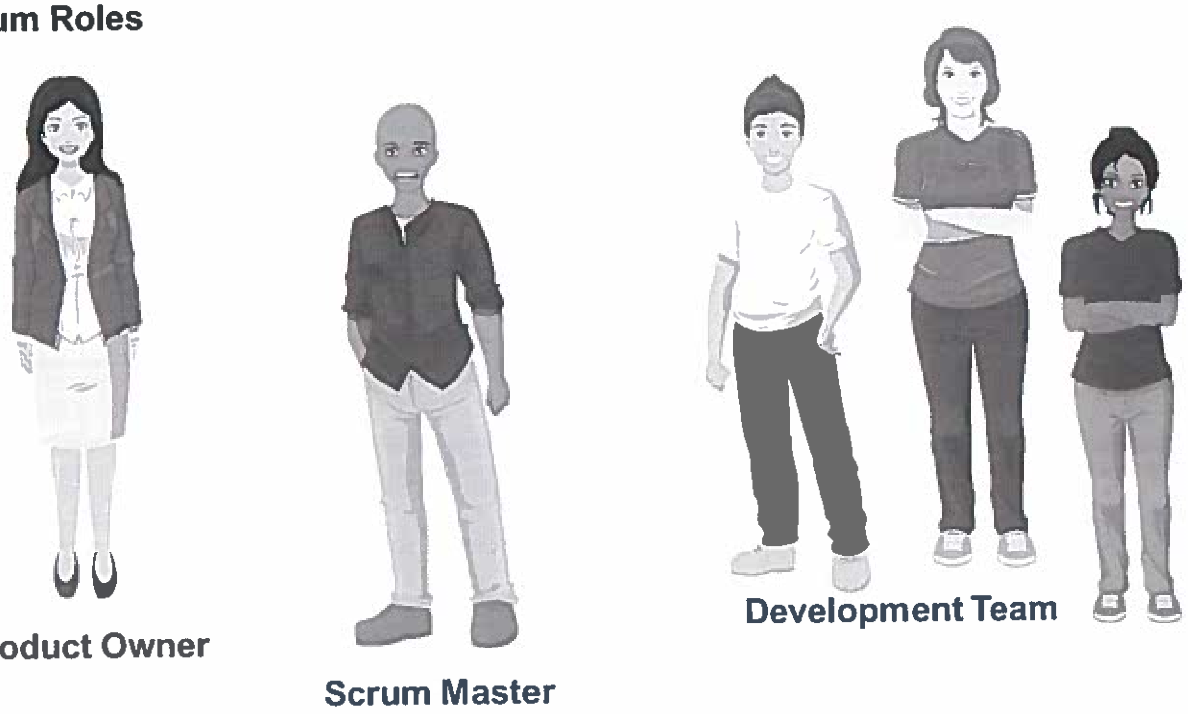
Assessment: Scrum

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



- 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.

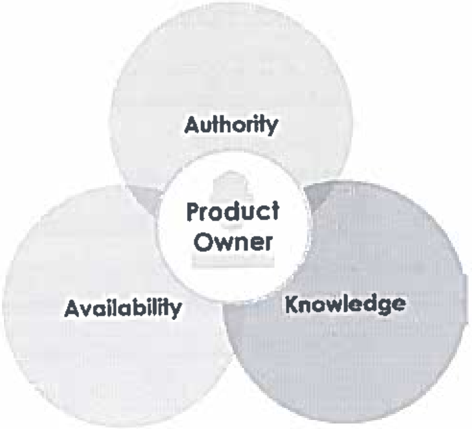
- The Product Owner role is one of the most critical roles in Scrum.
- Without this person communicating the Vision and providing the business priority for features, the Scrum Team does not have a direction for focusing on their work.
- We discuss the Product Owner role briefly in this course so that 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

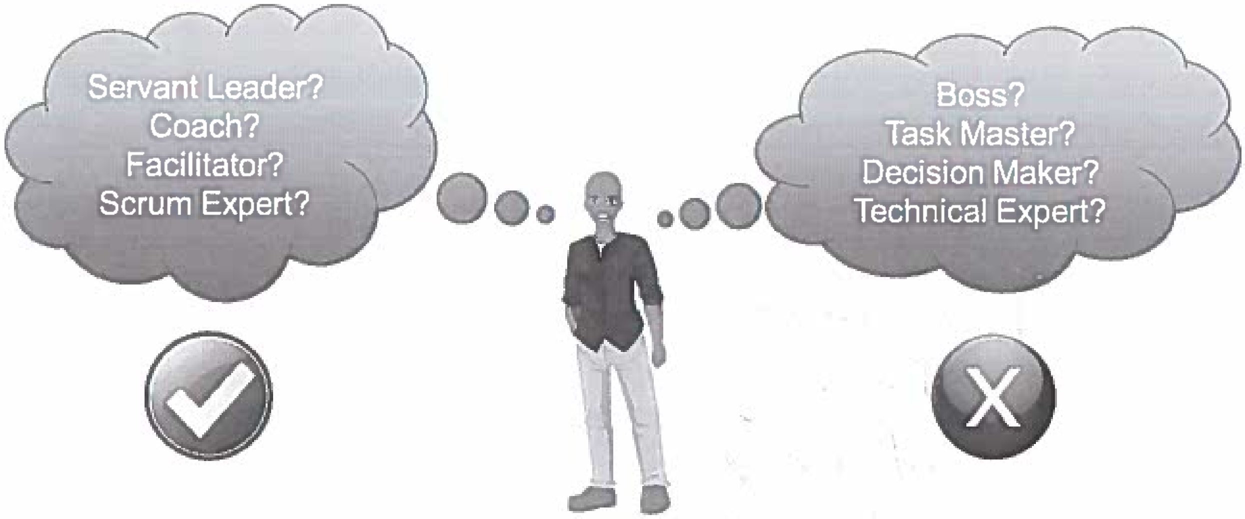
Questions:

- 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?

10 min

Tuckman
Forming
Stormy
Norming
performing

The Scrum Master

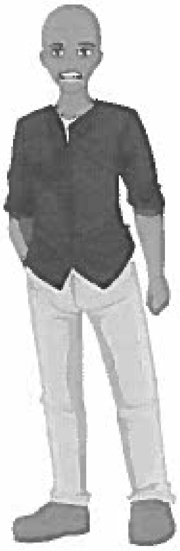


Why Not Agile Project Manager?

> "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?

10 min

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

10 min

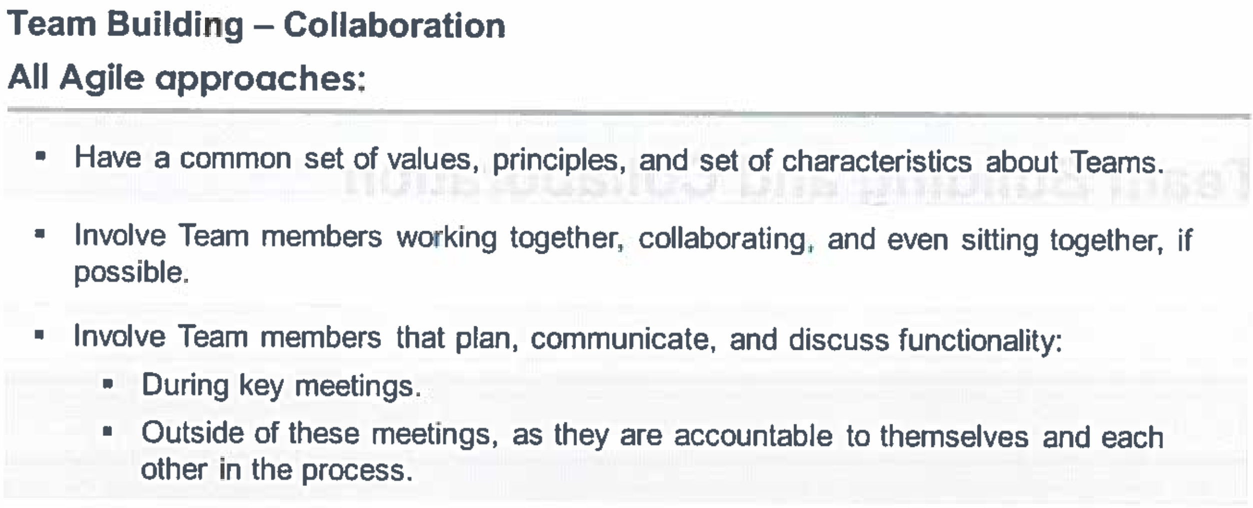
Question:

- 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.

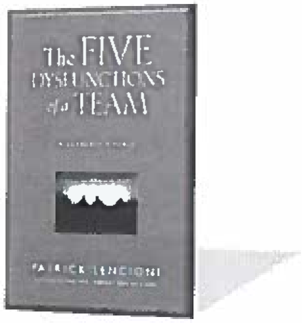
- Some people make the transition to Agile very easily, and others are not able to change as readily.
- Some of the frameworks, like Scrum and XP, have a role identified to help the Team members focus on adopting the process and to work together better as a Team.
- Other approaches do not necessarily 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.

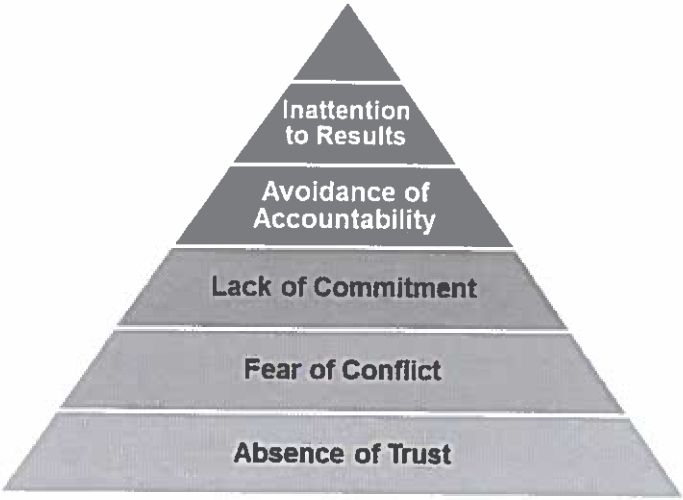
Team Building – Dysfunctions

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

10 min

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.

20 min

This page intentionally left blank.

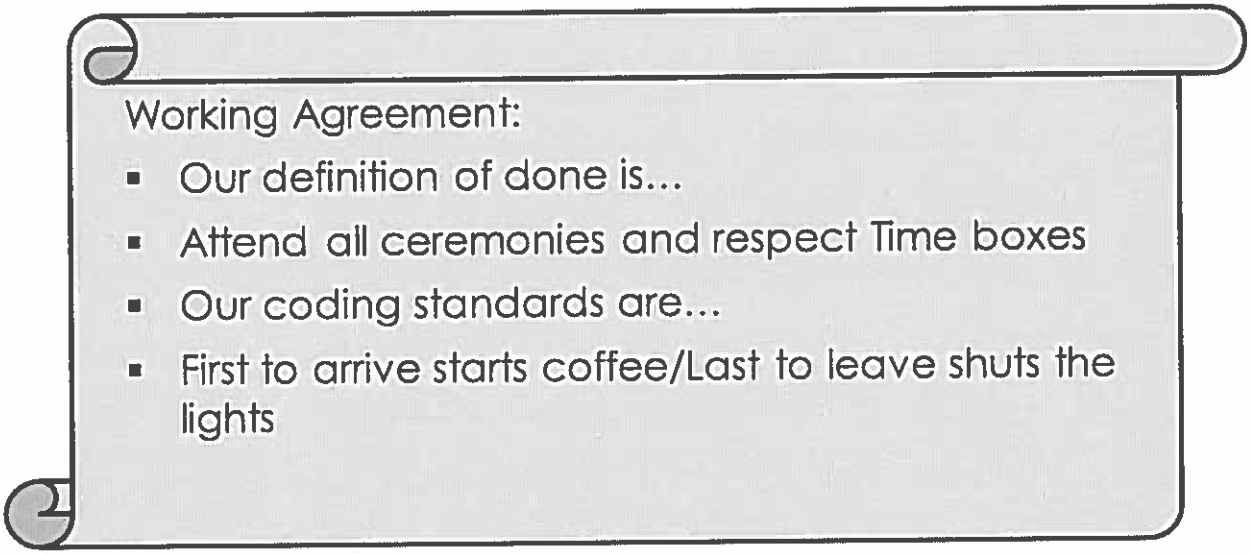
Working Agreements

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 Agreement:

- Our definition of done is…
- Attend all ceremonies and respect Time boxes
- Our coding standards are…
- First to arrive starts coffee/Last to leave shuts the lights

- 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.