SOFTWARE PROJECT MANAGEMENT METHODOLOGIES: AGILE AND KANBAN

Dr. SELİN METİN
Netaş Telekomünikasyon A.Ş.

Overview

- Lean and Agile Concepts
  - Value Added Work and Waste in Lean Principles
  - The Principles of Lean Software Development
  - Lean Values and Practices
- What is Agile?
  - Key Agile Concepts
  - Agile Project Techniques
  - Agile Team Goals
  - Waterfall versus Agile
  - Typical Challenges When Adopting Agile
  - Success Criteria for Agile Project
- What is Scrum?
  - Three Pillars of Scrum Theory
  - How Scrum Works
  - Scrum Roles: The Product Owner, The Development Team, The Scrum Master
  - Scrum Events
  - Product and Sprint Backlog
  - Definition of Done
- What Is Kanban?
  - Kanban Principles
  - The Kanban Board
  - Advantages and Disadvantages of Kanban
- Scrum vs. Kanban
- Evaluation Criteria to Determine the Best Approach for Your Project
- Comparison of Project Management Methodologies

Lean and Agile Concepts

- Lean is a “thinking principle” that focuses on the tasks and activities that are being conducted to complete the building of a product or a process to deliver a service. Additionally, the concepts of Lean include evaluating the flow of work to identify opportunities for improvement.
  - By conducting this analysis and changing the way we think about our work, we are able to identify the specific areas that need improvement.

- As we change the way we approach our work, and as we focus on increasing our efficiencies to deliver our projects to our stakeholders with higher quality products, it is helpful to focus on “Leaning out the waste.”
  - Waste ≈ “overhead” activities which have little value to the end product

- We want to reduce “Non-Value Add” activities and increase “Customer Value Add” activities.

Value Added Work in Lean Principles

- There are three types of work that we need to focus on as we evaluate our daily activities on a project:

- **Customer Value Added Work**:
  - Time directly spent on building the end product or deliverable
  - Agile teams prioritize this work and maximize their daily activities to be value add activities

- **Business Value Added Work**:
  - Any activities that we need to complete to maintain project management or governance compliance
  - Identify and simplify the ceremonial activities
  - Agile teams develop lighter yet just as effective artifacts to support the requirement
  - Try to replace some of the traditional project artifacts with empirical evidence (demonstrations and stand-ups versus status reports)

- **Non Value Added Work (Waste)**:
  - Time spent on work that is not directly related to building the end product.
  - “overhead” or “waste”

What is Waste?

- The pursuit of perfection is achieved by eliminating waste.
- The single word “waste” in English is described more richly with three Japanese terms:
  - Muda – literally meaning “waste” but implying non-value-added activity
  - Mura – meaning “unevenness” and interpreted as “variability in flow”
  - Muri – meaning “overburdening” or “unreasonableness”
- Perfection is pursued through the reduction of non-value-added activity but also through the smoothing of flow and the elimination of overburdening.
- In addition, the Toyota approach was based in a foundational respect for people and heavily influenced by quality assurance and statistical process control.

Lean Software Development

- Lean software development (LSD) is a translation of lean manufacturing principles and practices to the software development domain. It is adapted from the Toyota Production System that was designed in 1940s.
  - A software development lifecycle process or a project management process could be said to be "lean" if it aligns with the values and the principles of Lean Software Development.
- Focuses the team on delivering **Value** to the customer, and on the efficiency of the "Value Stream," the mechanisms that deliver that Value.
- Lean uses the idea of work product being "pulled" via customer request.
- It focuses decision-making authority and ability on individuals and small teams
  - Research shows this to be faster and more efficient than hierarchical flow of control
  - Also concentrates on the efficiency of the use of team resources, trying to ensure that everyone is productive as much of the time as possible. It concentrates on concurrent work and the fewest possible intra-team workflow dependencies.
- Strongly recommends that automated unit tests be written at the same time the code is written.
- The organization using a Lean software development process could be said to be Lean if it exhibited only small amounts of waste in all three forms ("mura," "muri," and "muda") and could be shown to be optimizing the delivery of value through effective management of risk.

The Principles of Lean Software Development

- Eliminate waste.
  - Regard any activity that does not directly add value to the finished product as waste.
  - Allow development teams to self organize and operate in a manner that reflects the work they're trying to accomplish.
- Build in quality.
  - Your process should not allow defects to occur in the first place. Do a bit of work, validate it, fix any issues that you find, and then iterate.
- Create knowledge.
  - Planning is useful, but learning is essential.
  - Promote strategies that help teams discover what stakeholders really want and act on that knowledge.
- Decide as late as possible (defer commitment).
  - Design flexible architectures that are change tolerant and by scheduling irreversible decisions to the last possible moment.
- Deliver quickly.
  - By limiting the work of a team to its capacity, which is reflected by the team's velocity, you can establish a reliable and repeatable flow of work.
  - Constraining teams to delivering potentially shippable solutions regularly motivates them to stay focused on continuously adding value.
- Respect people (empower the team).
  - Sustainable advantage is gained from engaged, thinking people.
- Optimize the whole.
  - Look at the bigger picture.

Lean Values

- Accept the human condition
  - Knowledge work is undertaken by human beings. Humans are emotional, social, and tribal, and our behavior changes with fatigue and stress. Embrace and accommodate the human condition rather than denying it and assuming logical, machine-like behavior.
- Accept that complexity & uncertainty are natural to knowledge work
  - There is chance or randomness at many levels (e.g. customers and markets, workers, defects and required rework are unpredictable). The purpose, goals, and scope of projects tend to change while they are being delivered.
  - Some of uncertainty and variability is knowable and its risks managed, but some variability is unknowable in advance and cannot be adequately anticipated. Thus systems of Lean Software Development must be able to react to unfolding events, and the system must be able to adapt to changing circumstances.
- Work towards a better Economic Outcome
  - Better economic outcomes will involve delivery of more value to the customer, at lower cost, while managing the capital deployed by the investors or owners in the most effective way possible.
- While enabling a better Sociological Outcome
  - Creating a workplace that respects people by accepting the human condition and provides systems of work that respect the psychological and sociological nature of people is essential.
- Seek, embrace & question ideas from a wide range of disciplines
- A values-based community enhances the speed & depth of positive change

Lean Practices - 1

- Lean Software Development does not prescribe any practices, but some activities have become common.
- Lean organizations seek to encourage **kaizen** through visualization of workflow and work-in-progress and through an understanding of the dynamics of flow and the factors (such as bottlenecks, non-instant availability, variability, and waste) that affect it.
- Process improvements are suggested and justified as ways to reduce sources of variability, eliminate waste, improve flow, or improve value delivery or risk management.
  - Lean Software Development processes will always be evolving and uniquely tailored to the organization within which they evolve.
  - It will also be unlikely that returning to an organization after a few weeks or months to find the process in use to be the same as was observed earlier. It will always be evolving.
- Cumulative Flow Diagrams
  - Cumulative flow diagrams plot an area graph of cumulative work items in each state of a workflow.
  - A truly Lean process will show evenly distributed areas of color, smoothly rising at a steady pace. The picture will appear smooth without jagged steps or visible blocks of color.
  - Used to visualize the quantity of work-in-progress at any given step in the work item lifecycle. This can be used to detect bottlenecks and observe the effects of “mura” (variability in flow).

Lean Practices - 2

- Visual Controls
  - Lean Software Development teams use physical boards, or projections of electronic visualization systems, to visualize work and observe its flow.
  - Enable team members to self-organize to pick work and collaborate together without planning or specific management direction or intervention.
- Virtual Kanban Systems
  - It uses a system of physical cards to limit the quantity of work-in-progress at any given stage in the workflow.
- Small Batch Sizes / Single-piece Flow
  - Work is either undertaken in small batches, often referred to as “iterations” or “increments,” or that work items flow independently, referred to as “single-piece flow.”
  - Single-piece flow requires a sophisticated configuration management strategy to enable completed work to be delivered while incomplete work is not released accidentally. This is typically achieved using branching strategies in the version control system.
  - A small batch of work would typically be considered a batch that can be undertaken by a small team of 8 people or less in under 2 weeks.
  - Frequent interaction with business owners to replenish the backlog or queue or work and a capability to release frequently.

Lean Practices - 3

- Automation
  - A high level of automation to economically enable single-piece flow and to encourage high quality and the reduction of failure demand (e.g. the use of automated testing, automated deployment, and software factories to automate the deployment of design patterns and creation of repetitive low variability sections of source code)
- Kaizen Events
  - **Kaizen means "continuous improvement"** and a kaizen event is the act of making a change to a process or tool that hopefully results in an improvement.
  - Stimulate a conversation about problems that adversely affect capability and, consequently, ability to deliver against demand. The essence of kaizen in knowledge work is that we must provoke conversations about problems across groups of people from different teams and with different skills.
- Daily standup meetings
  - Team discusses the dynamics of flow and factors affecting the flow of work. Particular focus is made to externally blocked work and work delayed due to bugs. Problems with the process often become evident over a series of standup meetings.

Lean Practices - 4

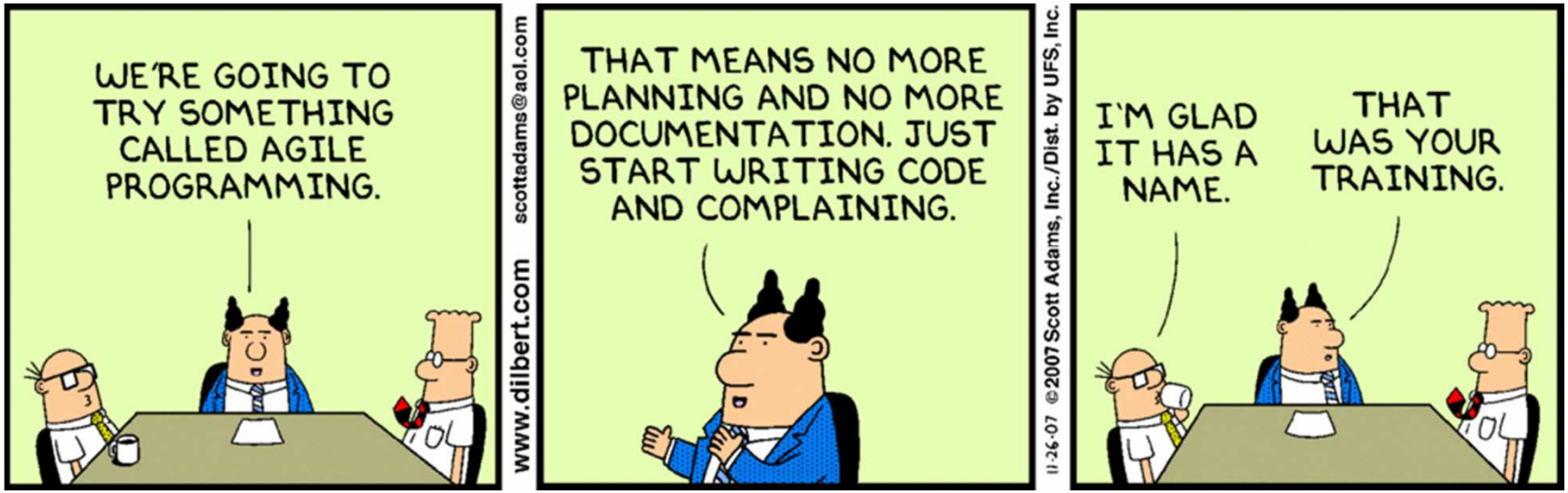
- Retrospectives
  - Project teams may schedule regular meetings to reflect on recent performance often after specific project deliverables or sprints are complete.
  - Asking questions like “what went well?”, “what would we do differently?”, and “what should we stop doing?”
  - Retrospectives typically produce a backlog of suggestions for kaizen events. The team may then prioritize some of these for implementation.

- Operations Reviews
  - An operations review is typically larger than a retrospective and includes representatives from a whole value stream. Operations reviews are typically held monthly.
  - The key differences between an operations review and a retrospective is that operations reviews span a wider set of functions, typically span a portfolio of projects and other initiatives, and use objective, quantitative data. Retrospectives, in comparison, tend to be scoped to a single project; involve just a few teams such as analysis, development, and test; and are generally anecdotal in nature.
  - An operations review will provoke discussions about the dynamics affecting performance between teams and propose changes.
  - Operations reviews typically produce a small backlog of potential kaizen events that can be prioritized and scheduled for future implementation.

What is Agile?

- Agile is a project methodology and approach that is derived using Lean thinking.
- Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches.
  - An agile team delivers work in small, but consumable, increments.
  - Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.
- Agile calls for collaborative cross-functional teams.
  - Open communication, collaboration, adaptation, and trust amongst team members are at the heart of agile. Although the project lead or product owner typically prioritizes the work to be delivered, the team takes the lead on deciding how the work will get done, self-organizing around granular tasks and assignments.
- Agile processes can empower product teams to significantly reduce the overall delivery risk vs. traditional process definitions.
- Agile isn't defined by a set of ceremonies or specific development techniques. Rather, agile is a group of methodologies that demonstrate a commitment to tight feedback cycles and continuous improvement.

Agile is not an unordered world!



Key Agile Concepts

- Agile encourages the following key concepts:
  - Frequent inspection and adaptation
  - A leadership philosophy that encourages team work, self-organization, and accountability
  - A set of engineering best practices that allow for rapid delivery of high-quality projects
  - A business approach that aligns development with customer needs and company goals

- Some of the foundations of agile include the following:
  - **Empiricism** – Ability to perform, stop, reflect, improve, and continue in a step-by-step process in efforts to increase productivity
  - **Prioritization** – Deliver work based on value to the business
  - **Self-Organization** – The team knows best how to deliver the work based on the resources and constraints
  - **Time-Boxing** – The team is required to complete the assigned tasks within the defined timelines.
  - **Collaboration** – The team commits to delivering the final products within the given timelines, which will encourage cross-team collaboration and ingenuity in completing the tasks.

Why Choose Agile?

- Teams can respond to changes in the marketplace or feedback from customers quickly.
  - "Just enough" planning and shipping in small, frequent increments lets the team gather feedback on each change and integrate it into future plans at minimal cost.
- Authentic human interactions are more important than rigid processes.
  - Collaborating with customers and teammates is more important than predefined arrangements.
  - Delivering a working solution to the customer's problem is more important than hyper-detailed documentation.
- An agile team unites under a shared vision, then brings it to life the way they know is best.
  - Each team sets their own standards for quality, usability, and completeness.
  - "Definition of done" informs how fast they'll churn the work out.
  - When you trust an agile team, the team feels a greater sense of ownership and rises to meet (or exceed) management's expectations.

Key Aspects of the Agile Project

- The Overall Team – Team, Scrum Master, product owner
- Product Backlog – the ongoing, prioritized list of “to do” tasks and features that are defined by business customers
- Sprint Planning and Backlog – planning session at the beginning of a sprint to determine the items that the team agrees to take on and deliver as the output of a sprint
- Stories/Story Board – collection of collective elements, functions, or “features” that are to be delivered within a sprint
- Daily Standup – daily discussion on what was accomplished, what remains to be done, and obstacles
- Sprint Burndown Chart – artifact showing progress and work remaining in a sprint
- Sprint Demonstration – demonstration conducted at the end of the sprint to show product(s) delivered
- Retrospective – team discussion following the sprint to identify successes and improvement opportunities

Agile Project Techniques

- A few techniques typically used on agile projects and that directly contribute to accelerating the time to delivery and the increased quality of the product being delivered, include the following:
  - Frequent inspection of the product and adaptation to the changes and input during the project
  - Aligning development with customer needs and company goals
  - Co-location of resources to the same work space
  - Self-organization and accountability
  - Becoming a team player
  - Elimination of “waste” and “ceremony”
  - Empirical demonstration of results
  - Customer is always present
  - Product managers and owners
  - Focus on key planning events: product planning, release/feature planning, iteration planning, sprint review, and stand-ups

Key Agile Principles

- Continuous and frequent delivery of valuable functionality and services
- Empowered self directed and self reliant teams
- Requirements evolution
- Active cross functional involvement is imperative
- Collaborative & cooperative approach between all stakeholders
- Feature completion before starting the next
- Testing is integrated throughout the project lifecycle – test early and often
- Customer feedback
- Flexible process depending on Business strategies:
  - Selection of Cross Functional primes embedded in Dev Sprints
  - Cloud Ops Model vs Regional Deployment
  - Release Readiness options: CA/GA/LA/SA
  - Story, Sprint, and Release Definition of Done
  - Fixed release timeframes vs variable depending on market needs

Estimate, Report and Plan

- Need a way to see the team's progress to be able to plan for future work or sprints.

- Agile project estimating
  - Agile project estimating helps both scrum and kanban teams understand their capacity.
  - Kanban teams set their WIP limit for each state based on their previous experiences and team size. Scrum teams use project estimating to identify how much work can be done in a particular sprint.
  - Many estimating techniques like planning poker, ideal hours, or story points exist to determine a numeric value for the task at hand.

- Agile reporting
  - Agile reports show the team's progress over time.
  - Project estimations come into play at the beginning and end of each sprint to determine what is done and to show how accurate the initial estimates are at the end. Agile reports, such as Burndown charts, show how many "story points" are completed during the sprint.

- Backlog management and grooming
  - A product backlog is a prioritized list of work for the development team to do that comes from product roadmap and its requirements. The development team pulls work from the product backlog for each sprint.
  - Grooming and maintaining the backlog helps teams achieve their long-term goals by continually adding and removing items based on the team's long-term capacity and changing business objectives.

Agile Team Goals

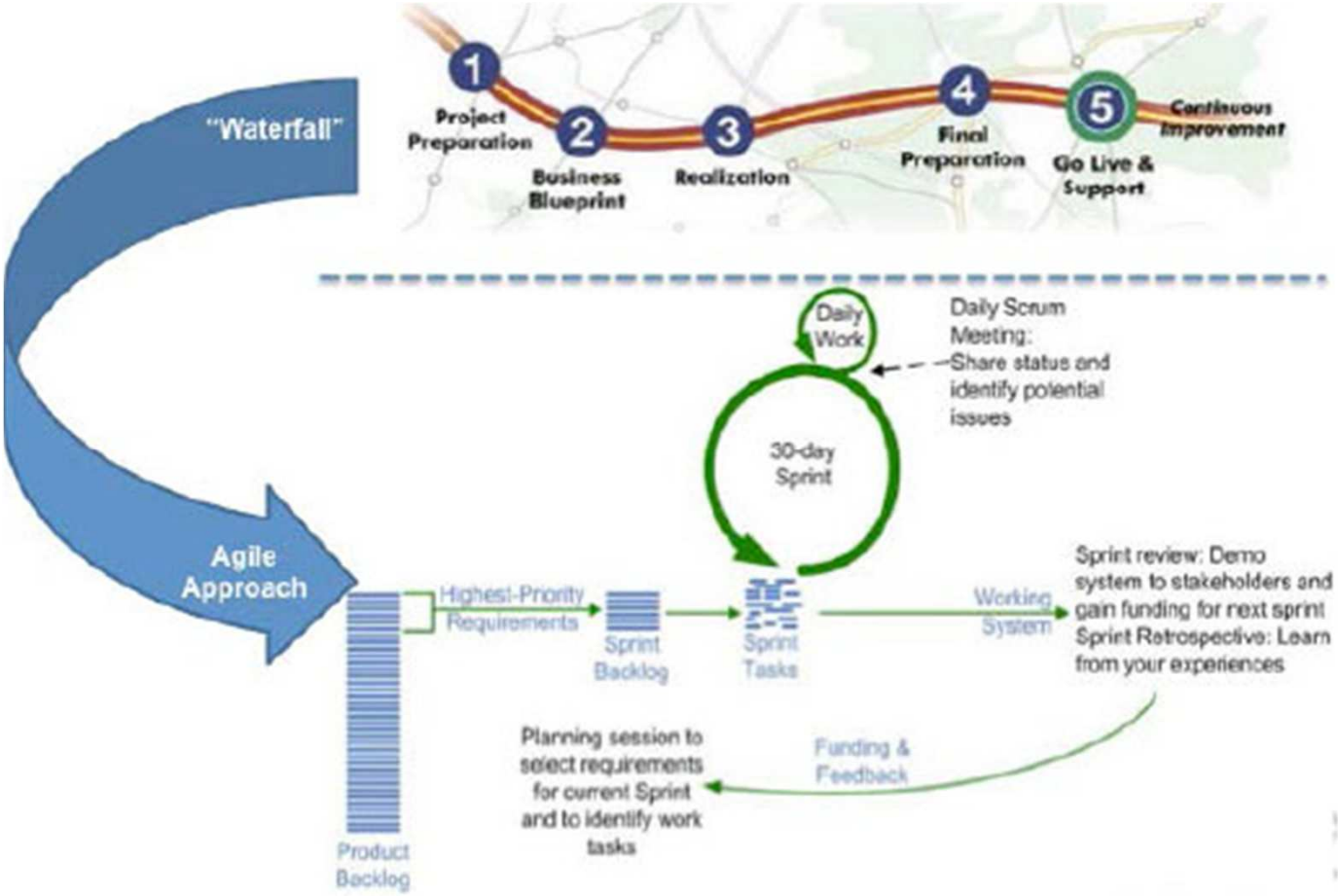
- Understanding pull principle and being innovative and productive
- Adopt practices, ceremonies and habits then evolve
- Find tools and frameworks that improves cooperation and helps automation
- Show people how the process works with teaching and coaching
- Strive to build a team to model the new behavior
- To achieve maximum speed and flexibility in competitive and changing environment
- Building a culture of openness and honest feedback
- Looking back at what is working well and what needs continuous improvement
- Continuous encouragement for collaboration and teamwork from the transformation leadership

Waterfall versus Agile

- There are some key differences between a traditional waterfall project and an agile project and these include the following:

Waterfall Project	Agile Project
Detailed, long-term project plans with single timeline	Shorter planning based on iterations and multiple deliveries
Definitive and rigid project management and team roles	Flexible, cross-functional team composition
Changes in deliverables are discouraged and costly	Changes in deliverables are expected and less impactful
Fully completed product delivered at the end of the timeline	Product delivered in functional stages
Contract-based approach to scope and requirements	Collaborative and interactive approach to requirements
Customer is typically involved only at the beginning and end of a project	Customer is involved throughout the sprint
Linear-phased approach creates dependencies	Concurrent approach seeks to reduce dependencies

Waterfall and Agile Project Approaches



Typical Challenges When Adopting Agile

- Agile is a new way to manage projects:
  - Makes all the typical “dysfunction” in a team or organization visible
  - Depending on how agile is introduced to a team, there may be resistance to the adoption and it may follow the mechanics but not the values of agile
- Agile is not right in all environments:
  - Spend time up front in assessing the environment and culture to determine agile readiness
  - If agile adoption is not managed well, the team will continue to deliver bad products but sooner, and doomed projects will fail faster.
- People are most comfortable with what they know:
  - Project team members are familiar with the phases and deliverables of the waterfall life cycle
  - Lack of incentive to increase speed to delivery
- Managing dependencies and sequencing of stories, tasks, and activities
- Management of large development objects (e.g., interfaces and conversion programs) in line with Sprint delivery

Success Criteria for Agile Project

- Conducting an agile readiness and cultural assessment before starting the project
- Identify the executive champion(s) – make sure they support this approach
  - Find the right product owner and engage stakeholders
- Establish “buy-in” to the agile process at all levels
- Start with something that can deliver a quick win – Establish confidence with the new approach in the organization
- The “art of storytelling” – be able to effectively define your requirements in the format of a story
- Continuously update the agile process and framework – Learn and adjust
  - Ability to remove impediments
  - Manage the flow of work
  - Be flexible in creating a model that works for the culture of the organization – Establish a process and framework that works best with your culture, resources, and environment
- Train the team – at all levels
- **Collaboration over co-location** –it is important to have the team collaborate; we all need to deal with the reality of distributed resources.
  - Focus on team work over mechanics
- Set rules of engagement for the team
- Select the right metrics and the right reporting tools
  - Align team and individual performance metrics to tangible project goals

Agile Project Management Approaches

- Traditional agile project management can be categorized into two frameworks: scrum and kanban. While scrum is focused on fixed-length project iterations, kanban is focused on continuous releases. Upon completion, the team immediately moves on to the next.

- Agile Scrum Methodology
  - Scrum is a lightweight agile project management framework with broad applicability for managing and controlling iterative and incremental projects of all types.
  - Simple, proven productivity, and ability to act as a wrapper for various engineering practices promoted by other agile methodologies.

- Lean and Kanban Software Development
  - The Kanban Method is used by organizations to manage the creation of products with an emphasis on continual delivery while not overburdening the development team. Kanban is a process designed to help teams work together more effectively.
  - Kanban promotes continuous collaboration and encourages active, ongoing learning and improving by defining the best possible team workflow.

What is Scrum?

- Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
- Scrum depends on short delivery cycles and feedbacks.
  - The prioritized requirements are delivered first.
- Scrum is:
  - Lightweight
  - Simple to understand
  - Difficult to master
- Scrum makes clear the relative efficacy of your product management and work techniques so that you can continuously improve the product, the team, and the working environment.
- The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules.
- The essence of Scrum is a small team of people. The individual team is highly flexible and adaptive.

Three Pillars of Scrum Theory

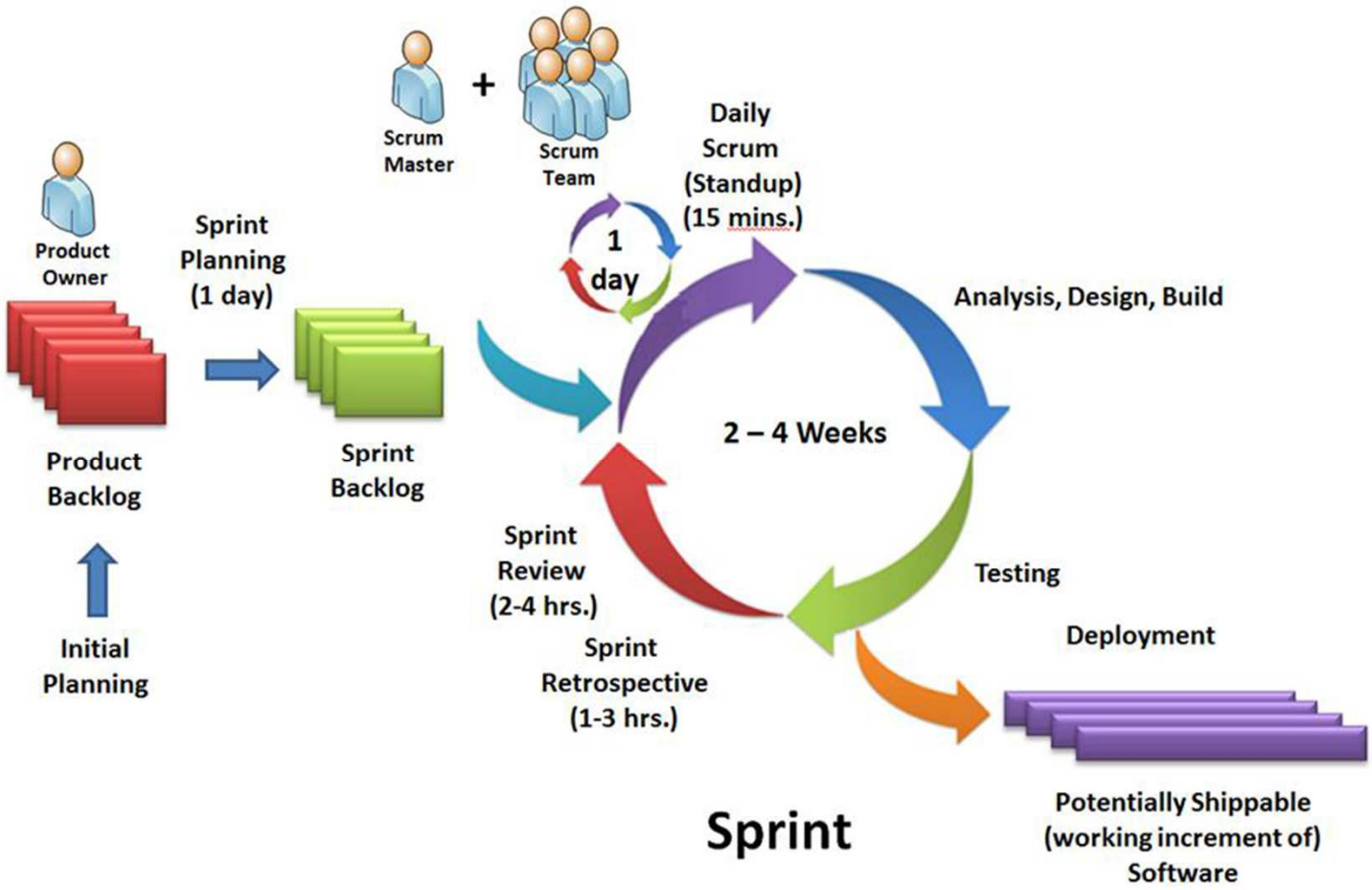
- Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known. Scrum employs an iterative, incremental approach to optimize predictability and control risk.

- **Transparency**
  - Significant aspects of the process must be visible to those responsible for the outcome. Aspects are defined by a common standard so observers share a common understanding of what is being seen. For example,
    - A common language referring to the process must be shared by all participants; and,
    - Those performing the work and those inspecting the resulting increment must share a common definition of “Done”.

- **Inspection**
  - Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal to detect undesirable variances. Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are most beneficial when diligently performed by skilled inspectors at the point of work.

- **Adaptation**
  - If an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted as soon as possible to minimize further deviation.

An Example Scrum Process

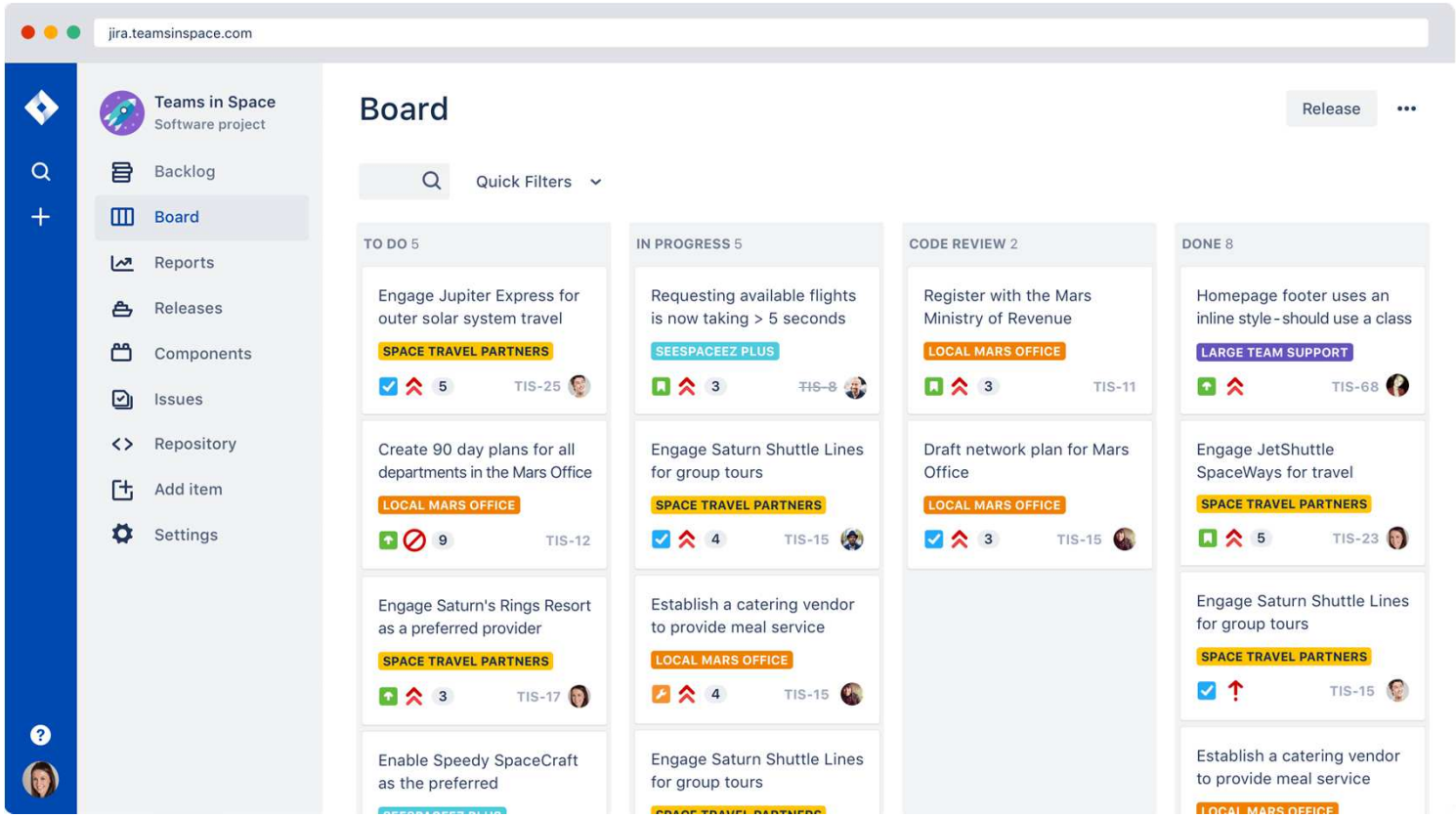


How Scrum Works

- Scrum uses fixed-length iterations of work, called **sprints**. There are four ceremonies that bring structure to each sprint.
- The “Product Owner” works closely with the team to identify and prioritize system functionality in form of a “Product Backlog”.
- The **backlog**, or body of work that needs to be done, consists of features, bug fixes, non-functional requirements, etc. – whatever needs to be done in order to successfully deliver a working software system.
  - Product backlog (owned by the product owner) which is a prioritized list of features
  - Sprint backlog which is filled by taking issues from the top of the product backlog until the capacity for the next sprint is reached
- Scrum teams have unique roles specific to their stake in the process. Typically there's a scrum master of the scrum method for the team; the product owner, who's the voice of the product; and the scrum team, who are often cross-functional team members in charge of getting things done.
- With priorities driven by the Product Owner, cross-functional teams estimate and sign-up to deliver “potentially shippable increments” of software during successive Sprints.
  - Once a Sprint’s Product Backlog is committed, no additional functionality can be added to the Sprint except by the team.
- Once a Sprint has been delivered, the Product Backlog is analyzed and reprioritized, if necessary, and the next set of functionality is selected for the next Sprint.

The Scrum Board

- Used to visualize all the work in a given sprint.
- During the sprint planning meeting, the team moves items from the product backlog into the sprint backlog.
- Scrum boards can have multiple steps visible in the workflow, like To Do, In Progress, and Done.
- Scrum boards are the key component for increasing transparency in agile project management.



The Scrum Team

- The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master.
- Scrum Teams are self-organizing and cross-functional.
  - Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team.
  - Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team.
- The team model in Scrum is designed to optimize flexibility, creativity, and productivity.
- Scrum Teams deliver products iteratively and incrementally, maximizing opportunities for feedback.
  - Incremental deliveries of “Done” product ensure a potentially useful version of working product is always available.

The Product Owner

- The Product Owner is responsible for maximizing the value of the product resulting from work of the Development Team.
- The Product Owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes:
  - Clearly expressing Product Backlog items;
  - Ordering the items in the Product Backlog to best achieve goals and missions;
  - Optimizing the value of the work the Development Team performs;
  - Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and,
  - Ensuring the Development Team understands items in the Product Backlog to the level needed.
- The Product Owner is one person, not a committee.
  - The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a Product Backlog item's priority must address the Product Owner.
- For the Product Owner to succeed, the entire organization must respect his or her decisions.
  - The Product Owner's decisions are visible in the content and ordering of the Product Backlog. No one can force the Development Team to work from a different set of requirements.

The Development Team

- They consist of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint.
- They are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team's overall efficiency and effectiveness.
- Development Teams have the following characteristics:
  - They are self-organizing;
  - Development Teams are cross-functional, with all the skills as a team necessary to create a product Increment;
  - Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person;
  - Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed.
- Development Team Size
  - Optimal Development Team size is small enough to remain nimble and large enough to complete significant work within a Sprint.
  - Fewer than three Development Team members decrease interaction and results in smaller productivity gains. Smaller Development Teams may encounter skill constraints during the Sprint.
  - Having more than nine members requires too much coordination. Large Development Teams generate too much complexity for an empirical process to be useful.
  - The Product Owner and Scrum Master roles are not included unless they are also executing the work of the Sprint Backlog.

The Scrum Master

- The Scrum Master is responsible for promoting and supporting Scrum by helping everyone understand Scrum theory, practices, rules, and values.
- The Scrum Master is a **servant-leader** for the Scrum Team.
  - Helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't and helps everyone change to maximize the value created by the Scrum Team.
  - Facilitates Scrum events as requested or needed.
- Scrum Master Service to the Product Owner
  - Ensuring that goals, scope, and product domain are understood by everyone on the Scrum Team as well as possible;
  - Finding techniques for effective Product Backlog management;
  - Understanding product planning in an empirical environment and practicing agility.
- Scrum Master Service to the Development Team
  - Coaching the Development Team in self-organization and cross-functionality;
  - Helping the Development Team to create high-value products;
  - Removing impediments to the Development Team's progress;
  - Coaching the Development Team in organizational environments in which Scrum is not yet fully adopted and understood.
- Scrum Master Service to the Organization
  - Leading and coaching the organization in its Scrum adoption;
  - Planning Scrum implementations within the organization;
  - Helping employees and stakeholders understand and enact Scrum and empirical product development;
  - Causing change that increases the productivity of the Scrum Team; and,
  - Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization.

The Four Ceremonies of Scrum

- Scrum prescribes four formal events for inspection and adaptation.

Sprint Planning	Sprint Demo	Daily Stand-up	Retrospective
A team planning meeting that determines what to complete in the coming sprint.	A sharing meeting where the team shows what they've shipped in that sprint.	Also known as a stand-up, a 15-minute mini-meeting for the software team to sync.	A review of what did and didn't go well with actions to make the next sprint better.

Scrum Events

- Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum.
- All events are time-boxed events – they have a maximum duration.
  - Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened.
  - The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process.
- Other than the Sprint itself, which is a container for all other events, each event in Scrum is a formal opportunity to inspect and adapt something.
- These events are specifically designed to enable critical transparency and inspection.

The Sprint

- The heart of Scrum is a Sprint, a time-box of during which a “Done”, useable, and potentially releasable product Increment is created.
  - Sprints have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.
- Sprints contain and consist of the Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective.
- During the Sprint:
  - No changes are made that would endanger the Sprint Goal;
  - Quality goals do not decrease; and,
  - Scope may be clarified and re-negotiated between the Product Owner and Development Team as more is learned.
- Each Sprint may be considered a project with no more than a one-month horizon. Like projects, each Sprint has a goal of what is to be built, a design and flexible plan that will guide building it, the work, and the resultant product increment.
  - When a Sprint’s horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase.
  - Sprints enable predictability by ensuring inspection and adaptation of progress toward a Sprint Goal at least every calendar month. Sprints also limit risk to one calendar month of cost.
- Sprint Goal
  - An objective set for the Sprint that can be met through the implementation of Product Backlog. It provides guidance to the Development Team on why it is building the Increment. It is created during the Sprint Planning meeting.
  - The Sprint Goal gives the Development Team some flexibility regarding the functionality implemented within the Sprint.
  - The selected Product Backlog items deliver one coherent function, which can be the Sprint Goal. The Sprint Goal can be any other coherence that causes the Development Team to work together rather than on separate initiatives.

Sprint Planning

- The work to be performed in the Sprint is planned at the Sprint Planning by the collaborative work of the entire Scrum Team.
- Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint.
- The Scrum Master ensures that the event takes place and that attendants understand its purpose.
- Sprint Planning answers the following:
  - What can be delivered in the Increment resulting from the upcoming Sprint?
    - The entire Scrum Team collaborates on understanding the work of the Sprint.
    - The input to this meeting is the Product Backlog, the latest product Increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team.
    - The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team.
    - During Sprint Planning the Scrum Team also crafts a Sprint Goal.
  - How will the work needed to deliver the Increment be achieved?
    - The Product Backlog items selected for this Sprint plus the plan for delivering them is called the Sprint Backlog.
    - The Development Team self-organizes to undertake the work in the Sprint Backlog, both during Sprint Planning and as needed throughout the Sprint.
    - The Product Owner can help to clarify the selected Product Backlog items and make trade-offs. If the Development Team determines it has too much or too little work, it may renegotiate the selected Product Backlog items with the Product Owner. The Development Team may also invite other people to attend to provide technical or domain advice.

Daily Scrum

- The Daily Scrum is a 15-minute time-boxed event for the Development Team.
- The Daily Scrum is held every day of the Sprint, at the same time and place each day to reduce complexity.
- Development Team plans work for the next 24 hours.
  - The Development Team uses the Daily Scrum to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog. The Daily Scrum optimizes the probability that the Development Team will meet the Sprint Goal.
- An example of daily scrum structure:
  - What did I do yesterday that helped the Development Team meet the Sprint Goal?
  - What will I do today to help the Development Team meet the Sprint Goal?
  - Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?
- The Scrum Master ensures that the Development Team has the meeting, but the Development Team is responsible for conducting the Daily Scrum.
- The Daily Scrum is an internal meeting for the Development Team. If others are present, the Scrum Master ensures that they do not disrupt the meeting.
- Daily Scrums improve communications, eliminate other meetings, identify impediments to development for removal, highlight and promote quick decision-making, and improve the Development Team's level of knowledge. This is a key inspect and adapt meeting.

Sprint Review

- A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.
- During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint.
  - Attendees include the Scrum Team and key stakeholders invited by the Product Owner;
- This is an informal meeting and the presentation of the Increment is intended to elicit feedback.
- The Sprint Review includes the following elements:
  - The Product Owner explains what Product Backlog items have been “Done” and what has not been “Done”;
  - The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved, and demonstrates the work that it has “Done” and answers questions about the Increment;
  - The Product Owner discusses the Product Backlog as it stands and projects likely target dates based on progress to date;
  - The entire group collaborates on what to do next, so this provides valuable input to subsequent Sprint Planning;
  - Review of how the marketplace or potential use of the product might have changed, what is the most valuable thing to do next, review of the timeline, budget, and potential capabilities for the following releases of the product.
- The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.

Sprint Retrospective

- The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.
- The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning.
- The Scrum Master ensures that the meeting is positive and productive and participates as a peer team member in the meeting from the accountability over the Scrum process.
- The purpose of the Sprint Retrospective is to:
  - Inspect how the last Sprint went with regards to people, relationships, process, and tools;
  - Identify and order the major items that went well and potential improvements; and,
  - Create a plan for implementing improvements to the way the Scrum Team does its work.
- The Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of “Done”, if appropriate and not in conflict with product or organizational standards.
- By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint.

Product Backlog

- The Product Backlog is an ordered list of everything that is known to be needed in the product.
  - It lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases.
  - Product Backlog items have the attributes of a description, order, estimate, and value.
- The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.
- A Product Backlog is never complete. The Product Backlog is dynamic; it constantly changes to identify what the product needs to be appropriate, competitive, and useful.
  - Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog.
- Multiple Scrum Teams often work together on the same product. One Product Backlog is used to describe the upcoming work on the product.
- Product Backlog refinement is an ongoing process in which the Product Owner and the Development Team collaborate on the details of Product Backlog items.
- The Development Team is responsible for all estimates.
  - The Product Owner may influence the Development Team by helping it understand and select trade-offs, but the people who will perform the work make the final estimate.
- Monitoring Progress Toward Goals
  - At any point in time, the total work remaining to reach a goal can be summed. The Product Owner tracks and compares this amount with work remaining at previous Sprint Reviews to assess progress. This information is made transparent to all stakeholders.

Sprint Backlog

- The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal.
  - A forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a “Done” Increment.
- The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum.
  - As new work is required, the Development Team adds it to the Sprint Backlog.
  - As work is performed or completed, the estimated remaining work is updated.
  - When elements of the plan are deemed unnecessary, they are removed.
  - Only the Development Team can change its Sprint Backlog during a Sprint.
- Monitoring Sprint Progress
  - At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed. By tracking the remaining work throughout the Sprint, the Development Team can manage its progress.
- Increment
  - The Increment is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints.
  - At the end of a Sprint, the new Increment must be “Done” - it must be in useable condition and meet the Scrum Team’s definition of “Done” regardless of whether the Product Owner decides to release it.

Definition of Done (DoD)

- Everyone must understand what “Done” means to ensure transparency.
- Each Increment is additive to all prior Increments and thoroughly tested, ensuring that all Increments work together.
- Any one product or system should have a definition of “Done” that is a standard for any work done on it.
- At the start of a release the Release team will define and agree on the following definitions of done (DoD):
  - Story DoD
    - Completion of working deployable software for a user story.
  - Sprint DoD
    - Administrative / Sprint closure checklist
  - Controlled Availability (CA) DoD
    - Ready to deploy to for Customer Beta quality
  - Release DoD
    - Ready to deploy to production final/SHIP quality

Story – Definition of Done

- The DoD for a story is defined according to the make up of the scrum team members set at release start. The scrum team may include extended team members such as Operations, and Services to deliver end to end deliverables during the Sprint as required.
  - This list should be used as a guideline when creating the tasks for each story.
- The scrum team is accountable for completing the DoD during the Sprint for each committed story.

Criteria	Details
Completion of working potentially deployable software	Defintion of "deployable" and deployment sites/customer should be agreed at Sprint start
Code is fully developed, integrated, sourced into required streams. Code inspections are a best practice but optional	
Acceptance test cases defined, 100% automated and 100% executed	Automated test cases added to CI suite. Test cases that can not be automated are added to the manual regression suite if critical.
Acceptance test cases 100% passing or accepted resolution plan with stakeholders for any failures	All failures tracked with JIRAs and/or new stories as required.
Automated CI Regression Suite executed and passing in Clean environment	
Demonstrable to customers, sales, deployment and support teams.	Demos should include show acceptance criteria has been met
Functional/Black-box test plan and cases created, reviewed and ready to execute post Dev Sprint	Acceptance tests broken into Tier 1, Tier 2 and Tier 3 prior Release Ready declaration. (Black box testing)
Functional Description, Operational Support System, and Services/Delivery impacts documented	Updated during the grooming of for each story, but documented at EPIC level for complete Feature view.
Fully tested procedures and supporting tools available for performing any deployment, maintenance, upgrade, etc. operation actions	
Customer Documentation completed	

Sprint – Definition of Done

- The Scrum Master is accountable for ensuring the Sprint level deliverables are completed.

Criteria	Details
Summary deployment readiness/ acceptance critiera of all committed stories in Sprint	Should include testcase execution and pass rates / DoD completion for all Stories in Sprint
Plan for any remaining stories, tcs, tasks, defects as agreed with PO	(including refactoring of story to push remaining content out)
Sprint closed and burn down chart updated, velocity report provided	
Demos / Knowledge Transfers held and recorded for all delivered stories	
Retrospective held and procedures updated with any resulting changes	
Release level risks updated	
All new 3rd parties reviewed/approved	
Additional Services / Operations dependencies met as required.	
Software build posted in repository for required teams and/or customer as required	
If required as per agreement with Security team then Security Scans Completed and Gating Vulnerability Corrections resolved	Discussion / agreement needed with with Security team at Release Start
Intellectual Property Assessment completed for new content	
Test Strategy updated for Completed EPICs (as needed)	
Fully tested procedures and supporting tools available for performing any deployment, maintenance, upgrade, etc. operation actions	MOP will continuously be updated and approvals started within the sprint.
Customer Documentation completed	Customer documentation feedbacks will continiously be updated and approvals started within the sprint.
Feature must not negatively impact capacity/performance – traffic testing is done regularly as agreed to at start of sprint.	

Agile Scrum Release Definitions

- **Release**: A version of a product or solution comprised of a group of usable features or products that are placed into production or ready to be deployed into production.
- **Release Cycle**: Frequency at which a version of the product or solution is ready to be deployed into production.
- **Release Plan**: Defines all the steps needed to move working software to production, Includes the release goal, release target date, determination of number of development Sprints, the scope of the deployment Sprint, and prioritization of product backlog items that support the release.

Scrum Master – The Power of Scrum

What Is Kanban?

- Kanban is Japanese for “visual sign” or “card.” It is a visual framework used to implement Agile that shows what to produce, when to produce it, and how much to produce.
- It encourages small, incremental changes to your current system and does not require a certain set up or procedure (meaning, you could overlay Kanban on top of other existing workflows).
- Kanban was inspired by the Toyota Production System and Lean Manufacturing in the 1940s.
  - Toyota improved its engineering process by modeling it after how supermarkets stock shelves. Engineer Taiichi Ohno noticed that supermarkets stock just enough product to meet demand, optimizing the flow between the supermarket and customer. Inventory would only be restocked when there was empty space on the shelf (a visual cue). And because inventory matched consumption, the supermarket improved efficiency in inventory management.
- Toyota brought these same principles to its factory floors - ‘just-in-time’ (JIT) product system. Different teams would create a card (or Kanban) to communicate that they had extra capacity and were ready to pull more materials.
  - Because all requests for parts were pulled from the order, Kanban is sometimes called the “pull system.”
- In this context, development work-in-progress (WIP) takes the place of inventory, and new work can only be added when there is an “empty space” on the team’s visual Kanban board. Kanban matches the amount of WIP to the team’s capacity, improving flexibility, transparency, and output.

Kanban Principles

- Agile software development teams today are able to leverage these same JIT principles by matching the amount of work in progress (WIP) to the team's capacity. This gives teams more flexible planning options, faster output, clearer focus, and transparency throughout the development cycle.
  - This approach aims to manage work by balancing the demands with available capacity, and improving the handling of system level bottlenecks.

- Kanban is based on 3 basic principles:
  - Work items are visualized to give participants a view of progress and process, from start to finish usually via a Kanban board. Seeing all the items (workflow) in context of each other can be very informative
  - Limit the amount of work in progress (WIP): Work is pulled as capacity permits, rather than work being pushed into the process when requested. This helps balance the flow-based approach so teams don‘t start and commit to too much work at once
  - Enhance flow (continuously improve): when something is finished, the next highest thing from the backlog is pulled into play. The flow of work (the movement of work) throughout the Kanban board should be monitored and improved upon.

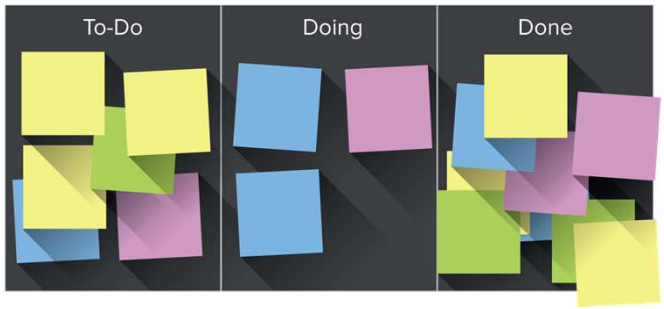
- In knowledge work and software development, this provides a visual process management system which aids decision-making about what, when and how much to produce.

About the Kanban Board

- A Kanban board is a tool to implement the Kanban method for projects.
  - Can be a physical board, with magnets, plastic chips, or sticky notes on a whiteboard to represent work items.
  - Project management software tools have created online Kanban boards.

- A Kanban board, whether it is physical or online, is made up of different swim lanes or columns. The simplest boards have three columns: to do, in progress, and done.
  - The columns for a software development project may consist of backlog, ready, coding, testing, approval, and done columns.

- Kanban cards (like sticky notes) represent the work and each card is placed on the board in the lane that represents the status of that work.
  - These cards communicate status at a glance.
  - You could also use different color cards to represent different details such as green cards for features and orange cards for tasks.



Advantages of Kanban

- Visual metrics: The Kanban board is easy to learn and understand, it improves flow of work, and minimizes cycle time. Two common reports are control charts and cumulative flow diagrams.
  - A control chart shows the cycle time for each issue as well as a rolling average for the team.
  - A cumulative flow diagram shows the number of issues in each state. The team can easily spot blockages by seeing the number of issues increase in any given state.
- Increases flexibility: Kanban is an evolving, fluid model. There are no set phase durations and priorities are reevaluated as new information comes in.
  - A kanban team is only focused on the active WIP. The product owner is free to reprioritize work in the backlog without disrupting the team, because any changes outside the current work items don't impact the team.
- Reduces waste: Kanban revolves around reducing waste, ensuring that teams don’t spend time doing work that isn’t needed or doing the wrong kind of work.
- Easy to understand: The visual nature of Kanban helps to make it incredibly intuitive and easy to learn. The team doesn’t need to learn a completely new methodology, and Kanban can be easily implemented on top of other systems in place.
- Improves delivery flow: Kanban teams optimize the flow of work out to customers. Kanban focuses on the just-in-time delivery of value and delivering work to customers on a regular cadence.
  - The faster a team can deliver innovation to market, the more competitive their product will be in the marketplace.
- Minimizes cycle time: Cycle time is the amount of time it takes for work to move through the team’s workflow. In Kanban projects, the entire team helps to ensure the work is moving quickly and successfully through the process.
  - When only one person holds a skill set, that person becomes a bottleneck in the workflow. Code review and mentoring help to spread knowledge so that team members can take on heterogeneous work, which further optimizes cycle time.

Disadvantages of Kanban

- Many of the disadvantages associated with Kanban come with misuse or mishandling of the Kanban board. An outdated or overcomplicated board can lead to confusion, inaccuracies, or miscommunication.

- Outdated board can lead to issues: The team must be committed to keeping the Kanban board up to date, otherwise they'll be working off inaccurate information. And once work is completed based off an out-of-date board, it can be hard to get things back on track.

- Teams can overcomplicate the board: The Kanban board should remain clear and easy to read, however some team members may learn "new tricks" they can apply to their board. Adding these kinds of bells and whistles to the Kanban board just buries the important information.

- Lack of timing: A frequent complaint about Kanban is that you don't know when things will be done. The columns on the Kanban board are only marked by phase (to do, in progress, complete), there are no timeframes associated with each phase, so you really don't know how long the to do phase could last.

Scrum vs. Kanban

- Some teams blend the ideals of kanban and scrum into "scrumban." They take fixed length sprints and roles from scrum and the focus on work in progress limits and cycle time from kanban.

	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Key metrics	Velocity	Cycle time
Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

Agile Process Examples

- Following charts display a set of agile process example flows
  - Release Planning Process: Development Scrum
  - Release Planning Process: Product Readiness
  - Sprint Planning Process
  - Anatomy of a Development Sprint
  - Test Driven Design – Foundation for Agile Quality
  - Project Phases Example
  - Agile Release Process - Overview

Release Planning Process: Development Scrum

Goal: Establish the overall release schedule and determine in what sprint stories will likely be delivered.

- Product Vision
- High level prioritized goals & roadmap
- Product Backlog (priority draft)
- Rough Estimates

- **Who**: Product Owner, Scrum Team, Scrum Master, Key Stakeholders
- **When**: before release n+1 begins (.5 -2 days)
- **How / Topic(s)**:
  - PO presents the vision, strategy and goals.
  - PO present key dates and milestones.
  - PO presents draft of the prioritized backlog.
  - Discussion to understand user stories.
  - Review rough estimates + prioritized features.
  - Agreement on Sprint length (in weeks) and target release dates.
  - Release Plan is organized by time (release every N sprints).

- Release Plan with target release date
- Selected stories for the release
- Prioritized product backlog
- Stakeholder consensus
- Key risks and assumptions
- Updated team capacity report

Release Planning Process: Product Readiness

Goal: Identify and create plan for non-functional deliverables required for delivery of the product

Product Vision

Release Plan

Product Readiness backlog (tasks)

- **Who**: Product Owner and Product Readiness Team - stakeholders from outside the scrum team who have tasks associated with releasing the product
- **When**: At Release Start (once features selected for release)
- **How / Topic(s)**:
  - PO presents the vision, strategy and goals.
  - PO presents prioritized features targeted for delivery
  - Product Readiness prime presents target release schedule
  - Review prioritized product readiness backlog
  - Review deployment plans and schedule for deployment to various production systems
  - Product readiness tasks identified along with target dates -> Product Readiness Plan

Product Readiness Release Plan

Prioritized product readiness backlog

Stakeholder consensus

Key risks and assumptions

Options for tracking product readiness tasks:

- *Epics/stories in Jira*
- *Tasks*

Sprint Planning Process

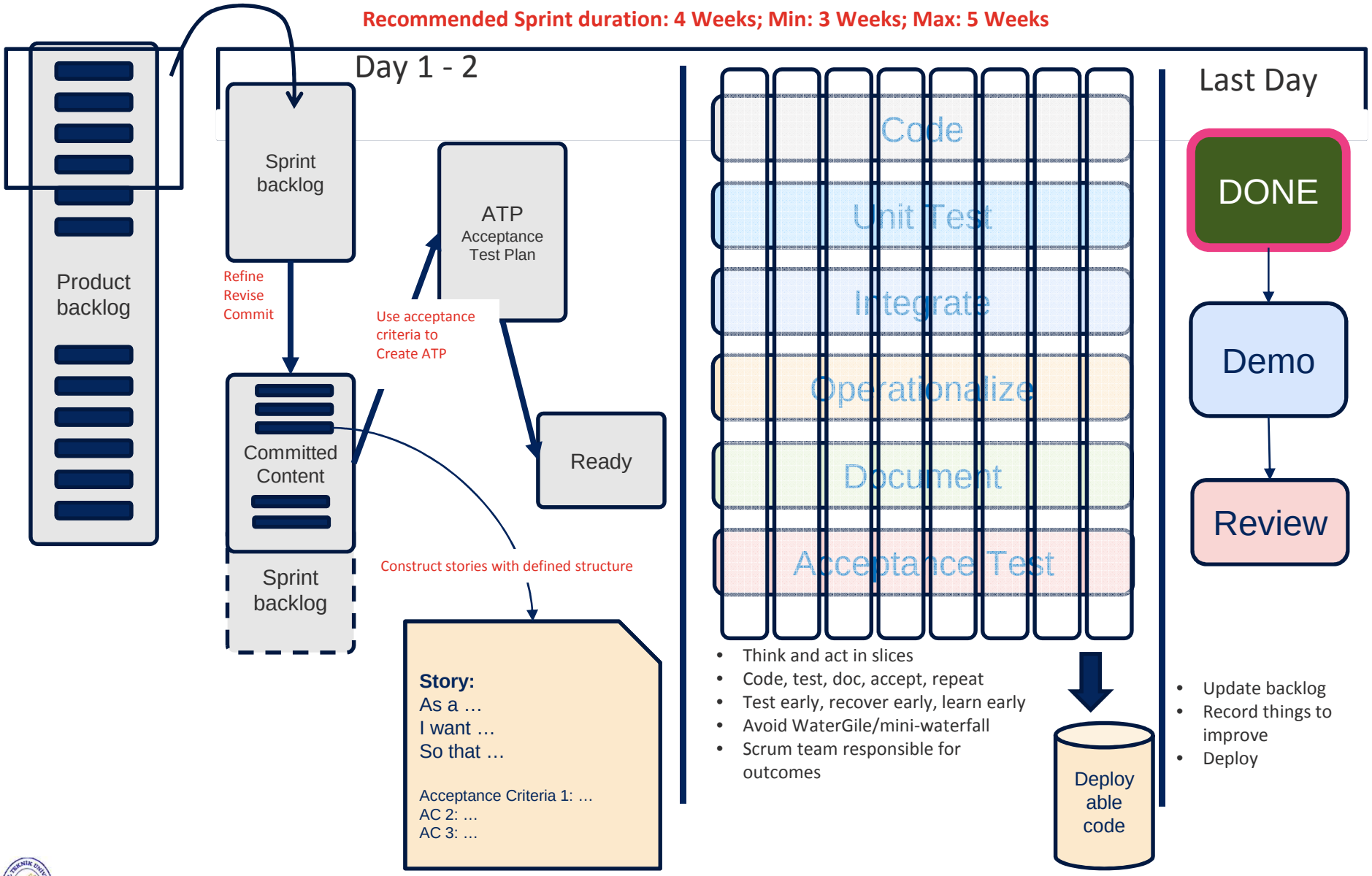
Goal: Team to plan and agree on backlog items they can complete and confirm the tasks required to support acceptance.

- Release Plan
- Prioritized product backlog
- Teams capabilities (hours)
- Prior Velocity
- Story Effort Estimation
- Schedule risks / Business conditions
- Retrospective Findings

- **Who**: Product Owner, Scrum Team, Scrum Master
- **When**: 1st day of each Sprint
- **How / Topic(s):**
  - PO presents the backlog items in priority order for review.
  - Stories with failed acceptance tests from prior sprints are added
  - Discuss story creation for defects from prior sprints
  - Review and clarify user stories.
  - Breakdown larger stories and each story into tasks (optional) and acceptance criteria.
  - Tasks are estimated in hours. (optional)
  - Developer and testers assigned to stories per available hours
  - Process continues until all available hours are used for the sprint.
  - Consider improvements from last Sprint for MoO and DoD updates

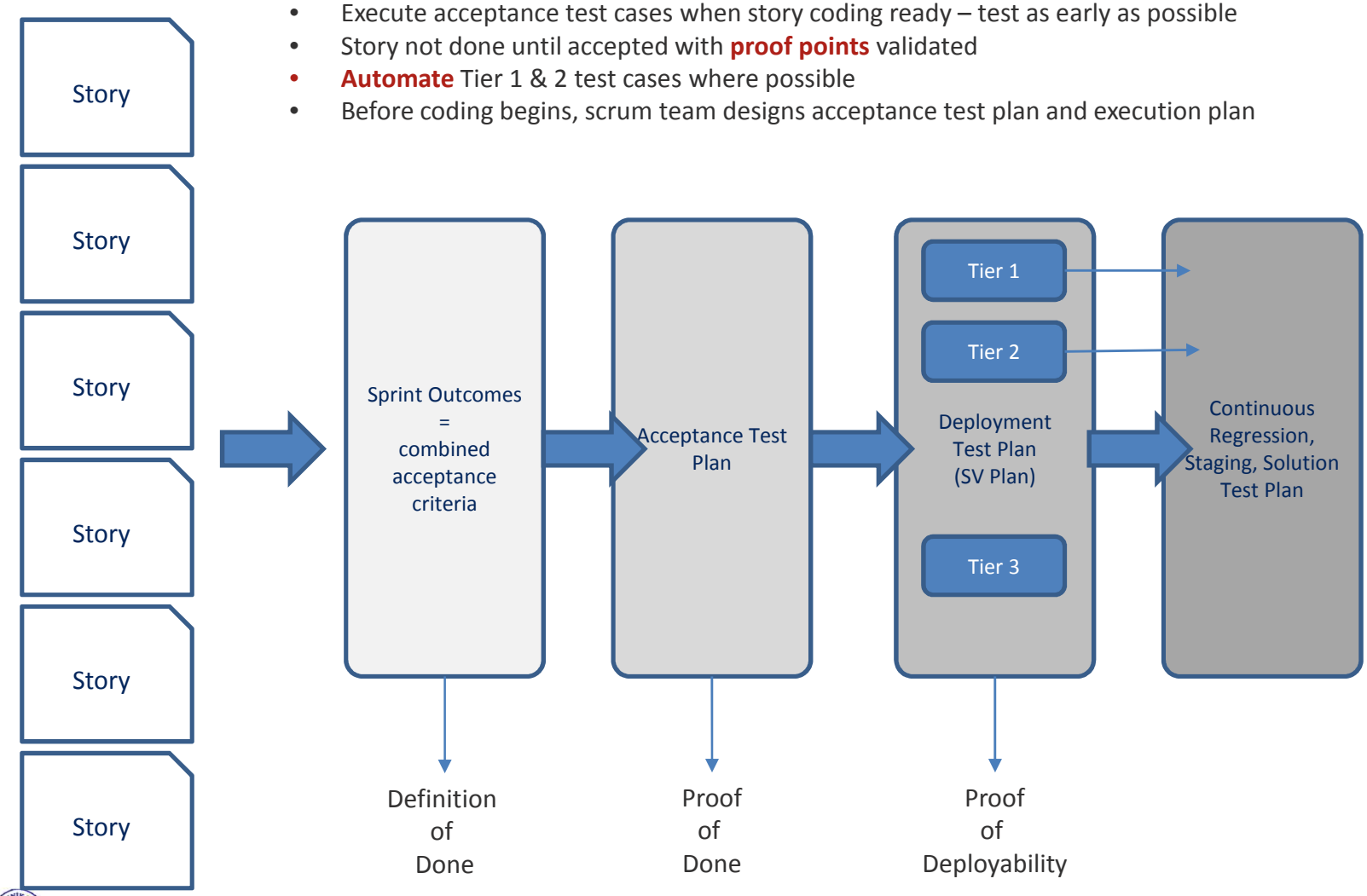
- Sprint Plan
- Selected stories for the sprint
- Key risks and assumptions
- Stakeholder consensus
- Updated MOO and DoD
- New 3rd parties reviewed with Legal
- Acceptance Criteria Defined

Anatomy of a Development Sprint



Test Driven Design – Foundation for Agile Quality

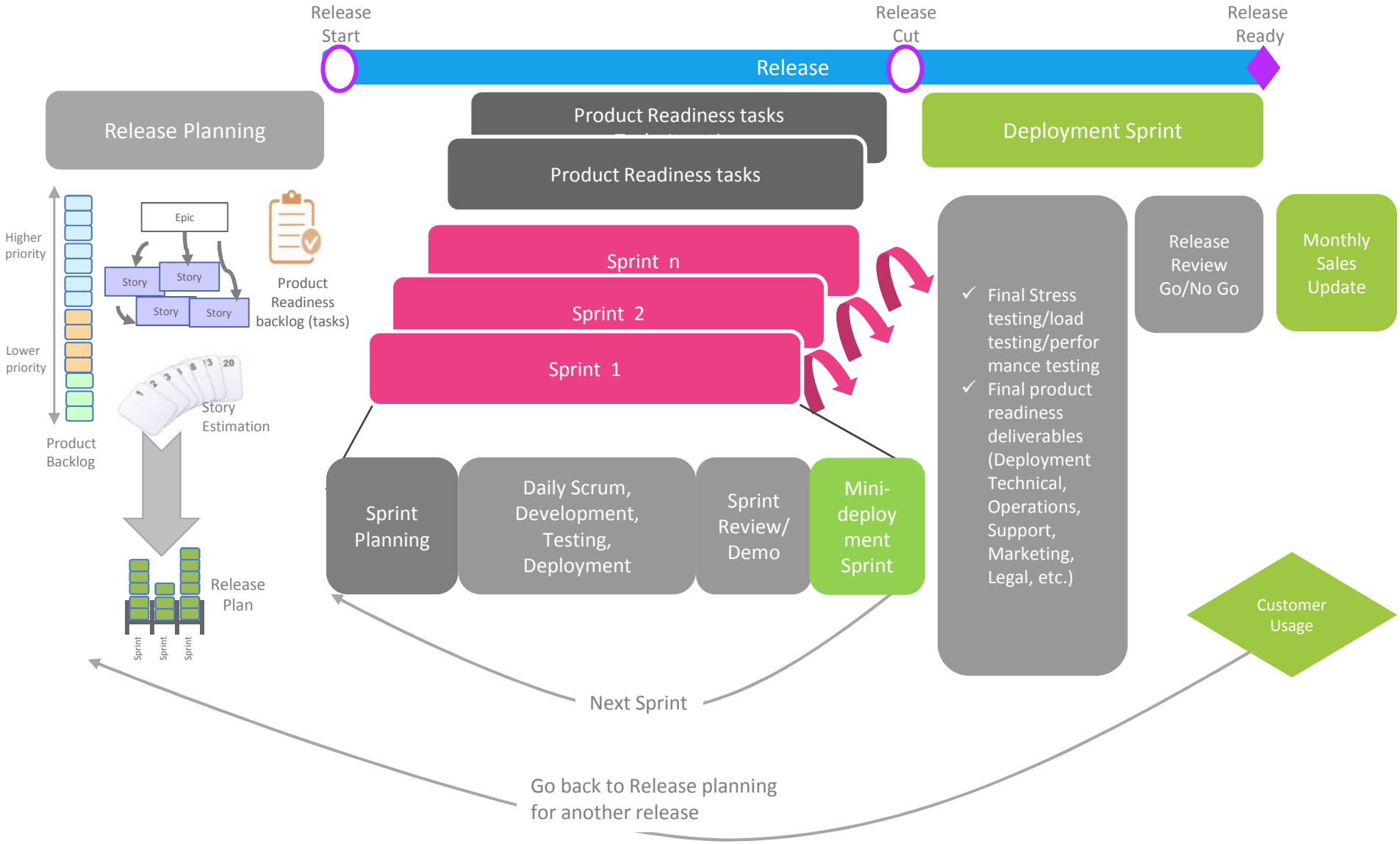
- **Scrum team** comprised of minimum of scrum master, design, test & product owner
  - Includes documentation, methods, operations, support where possible
- **Scrum team** responsible for delivering **outcomes** – tested, deployable software
- Execute acceptance test cases when story coding ready – test as early as possible
- Story not done until accepted with **proof points** validated
- **Automate** Tier 1 & 2 test cases where possible
- Before coding begins, scrum team designs acceptance test plan and execution plan



Project Phases Example

- Sample Agile project phases for a software product release

Agile Release Process - Overview



Evaluation Criteria to Determine the Best Approach for Your Project

- Project Characteristics:
  - Requirements – how rigid and defined are the requirements?
  - Effort/duration – How long is the planned project duration? >6 months, >12 months, >18 months?
  - Interfacing systems – How many interfacing systems are in scope? How complex are these interfaces?
  - Regulatory compliance – Are there any compliance requirements that provide restrictions or additional requirements for the project team?
  - Project inter-dependencies – How many other projects are running concurrently? What is the impact to the key decision-makers? Are they any overlaps with project resources?

- Sponsor Characteristics
  - Sponsor buy-in – Does the project have the right level of sponsorship? Are they committed to the mission of the project? Are the sponsors dedicated and willing to support the project as needed?
  - Periodic validation – Will the sponsor be available to participate in key validation sessions?

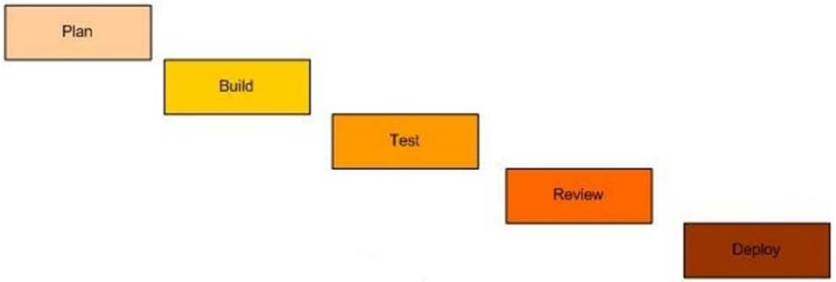
- Team Characteristics
  - Team size – How large is the team? Can it be broken down into teams of 8 to 12 people?
  - Resource dedication – Are key resources dedicated to the project? If not, can parameters around resource availability be established and supported?
  - Technology/business domain knowledge – How well do team members know the product being delivered? Is their domain expertise at the level that it will not impede the team's velocity?
  - Collaboration – Does the project environment foster collaboration? Are there tools in place to facilitate project team collaboration (e.g., video conferencing, shared document repositories, and so forth)
  - Co-location – How many of the team members are co-located? How many are distributed to different locations?

- Agile Awareness and Acceptance
  - Training at all levels – Has the organization committed to training the executive, project team, and subject matter experts in agile?
  - Ability to apply agile techniques for all aspects of the project – Is the project management team committed to the values of agile and to the techniques being used?

Comparison of Project Management Methodologies - 1

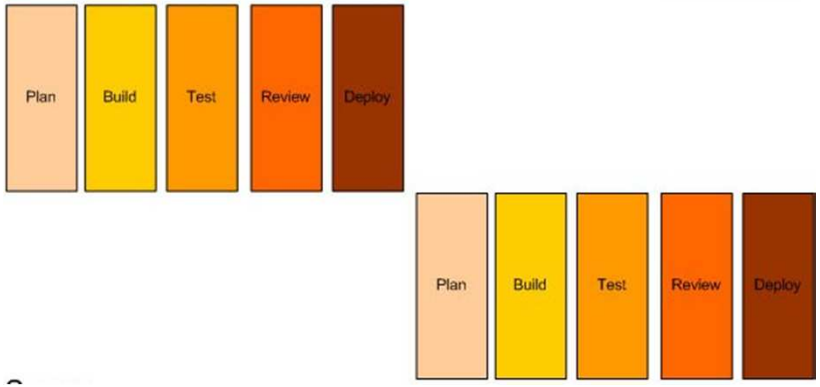
Waterfall Development

- You will rarely aim to re-visit a ‘phase’ once it’s completed. As such, you better get whatever you’re doing right the first time!



Iterative Waterfall Development

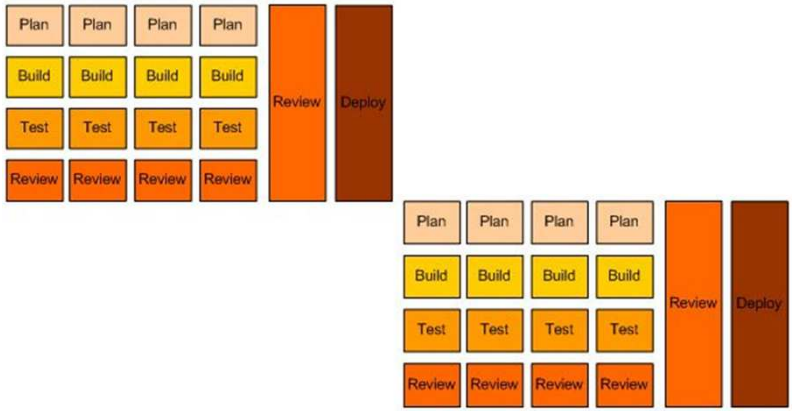
- Focus on delivering a sprint of work as opposed to a series of valuable/shippable features.
- You’re more or less forced to estimate each phase separately.
- The whole team must remain focused on delivering the end goal, not the separate phases.
- Velocity and burn downs are far less useful in this type of environment – you don’t benefit from early-warning-signs as you don’t find out whether you’re on track until the end of the sprint.



Comparison of Project Management Methodologies - 2

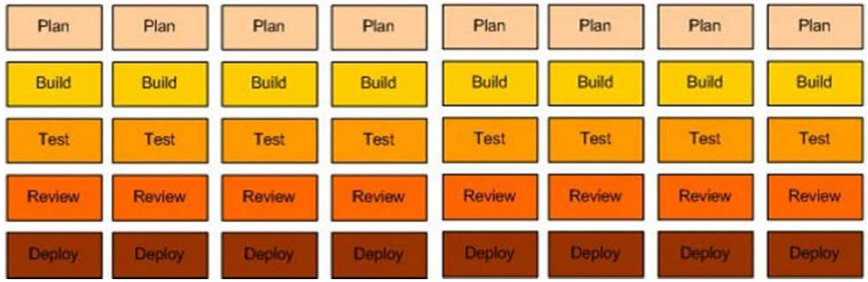
Scrum Development

- Focus on delivering fully-tested, independent, valuable, small features
- Diversify the risk – if one feature goes wrong, it should not impact another feature.
- Still plan the work in iterations and release at the end of each iteration.



Lean Development

- Focus on features as opposed to groups of features
- Select, plan develop, test and deploy one feature (in its simplest form) before you select, plan, develop, test and deploy the next feature.
- Further isolate risk to a feature-level.
- Aim to eliminate ‘waste’ wherever possible – therefore do nothing until you know it’s necessary or relevant.



Summary

- Lean Concepts: reduce “Non-Value Add” activities and increase “Customer Value Add” activities.
- Agile encourages the following key concepts:
  - Frequent inspection and adaptation
  - A leadership philosophy that encourages team work, self-organization, and accountability
  - A set of engineering best practices that allow for rapid delivery of high-quality projects
  - A business approach that aligns development with customer needs and company goals
- Foundations of agile: **Empiricism**, **Prioritization, Self-Organization**, **Time-Boxing, Collaboration**
  - Estimate, Report and Plan through backlog
- **Scrum** uses fixed-length **sprints**. The “Product Owner” works closely with the team to identify and prioritize system functionality in form of a “Product Backlog”, or body of work that needs to be done.
  - With priorities driven by the Product Owner, cross-functional teams estimate and sign-up to deliver “potentially shippable increments” of software during successive Sprints. Once a Sprint’s Product Backlog is committed, no additional functionality can be added to the Sprint.
  - The Scrum Team Roles: Product Owner, Development Team, Scrum Master
  - The Four Ceremonies of Scrum: Sprint Planning, Sprint Demo, Daily Standup, Retrospective
  - Any one product or system should have a definition of “Done” that is a standard for any work done on it.
- **Kanban principles**: Visual work items, Limit the amount of work in progress, Continuously improve
  - A Kanban board is made up of different swim lanes or columns. The simplest boards have three columns: to do, in progress, and done.
- Not all projects are a good fit for the agile approach. Assess each project with the evaluation criteria and make a decision based on what works best in the organization's environment.

Some References

- https://www.pmi.org/learning/library/agile-versus-waterfall-approach-erp-project-6300
- https://www.atlassian.com/agile/project-management
- https://www.scrumguides.org/scrum-guide.html#team
- The Scrum Guide, Ken Schwaber and Jeff Sutherland, 2017

Common Questions About Kanban - 1

Q: How do you organize meetings and maintain focus without a Scrum Master?

- Someone on the team needs to take initiative to put the meeting on the calendar and ensure the conversation stays on track. Even without a Scrum Master, it normally isn't too big of an issue.
- The Kanban board helps maintain focus during the meeting. During the meeting, you can go through the board from left to right and look for stories that have not moved since the last meeting. Instead of talking about accomplishments, you can just look at the cards on the board. The one question you do need to ask during a meeting is about the roadblockers or challenges to getting an item finished.
- You could also try a kaizen meeting, where you only invite people who are involved in the task at hand. Each person discusses problems and challenges, and how his or her job could be done more efficiently. Then, the whole group talks about solutions to those issues.
- Kaizen also can include a kaizen facilitator, who encourages the team to openly discuss critical issues.

Common Questions About Kanban - 2

Q: How can Kanban satisfy management's desire for predictable delivery?

- To some extent, Kanban trades predictability for efficiency. There are no timebox constraints or planning, however once a team has optimized the flow of work and can get a sense of how long certain tasks take, there will be some level of predictability.

- If management still needs more defined predictability (which is not the Kanban approach), you may need to try managing expectations. In a traditional model, you have a predictable date of delivery, but in reality, no one is going to deliver a product by that date if it's not complete. Management is always going to wait for the product to be complete, regardless of the original date set. In the Kanban model, the expectations need to be adjusted to focus on delivering the product when it's ready and complete.

Common Questions About Kanban - 3

Q: How do you use Kanban when you're on a deadline?

- There are a couple different ways you can handle deadlines in a Kanban model. You can simply write the deadlines on the Kanban cards, making sure these deadlines act more as guidelines rather than hard-and-fast due dates (in Kanban, you shouldn't sacrifice quality for timing).

- You could also change how you and your team approach deadlines. In Kanban's truest form, there is no need for them. The Kanban system will make sure that all tasks are completed as soon as possible, so a deadline is no longer necessary.

Q: Is WIP driven by resource availability?

- Yes. When setting WIP limits, you need to look at how many people you have on your team and how many tasks you want them to work on at the same time.

Common Questions About Kanban - 4

Q: Can Kanban be used for other projects besides software development?

- Yes, Kanban can improve process results, reduce production times, and help manage workflow in almost any industry. For example, in the game development industry, Kanban helps shorten the video process timeline and reduce waste. In real estate, it brings more efficiency by tracking contracts, prospects, and listings on various boards. And in finance, Kanban can quickly identify bottlenecks and increase speed-to-market.

Q: How do you know if the WIP limit is correct?

- There is no formula for setting the right WIP limits. It's very common for limits to be wrong in the beginning, but you just need to adjust them as the project progresses. A good place to start is 1.5 for available resource, but you should constantly be reevaluating this number and making changes as necessary.