



Business Analysis & Project Management in Agile Software Development - Critical Links

**Prepared By – Brian Basu, IIBA Austin Chapter
[Not for Redistribution]**



Learning Objectives

- Acquire a holistic view of Agile Methodologies
- Review Agile Values and Principles
- Understand the Agile Project Management Model
- Deep dive into scoping for agile projects and requirements management using agile tools
- Deep dive into important elements of scrum
- Be a leader on agile projects
- Measure success of agile projects

Holistic view of Agile Methodologies





Development models - Defined and linear

- Highly defined models are based on prediction. They are also called linear models.
- In defined models, processes are planned in detail before they're performed.
- A defined model is linear when the steps it describes are taken in sequence. Usually it is difficult to revisit a step to make a change.
- A project manager using this type of model begins a project by understanding and documenting, or defining, each of its phases. This becomes the foundation for all project activities. It will be hard to change the plans, once the project is in development because it would involve revisiting completed steps of the process.
 - The waterfall model is an example of a highly defined model.



Development Models - Empirical and iterative

- An empirical approach involves frequent evaluation and adaptation, with the flexibility to change based on new information.
- Not all aspects of a project may be clearly defined at the start because customers may not be able or want to outline their requirements too far into the future.
- They can respond to changes in customer and market requirements because they incorporate continual inspection and adjustment into the development cycle.
- Also, customers are encouraged to participate actively in the development process through regular feedback.
 - The Agile Model is an example of a highly empirical model.
- Although every agile methodology has different characteristics, they all maintain essential agile principles.
- Three widely used agile methodologies are Scrum, Extreme Programming – also known as XP, and Lean development.

Scrum

➤ Key elements of Scrum:

- **Product Backlog** - The product backlog is the only document used to list all possible requirements during a project, and because it's constantly updated and reordered, it's considered to be complete.
- **Sprint backlog** - A sprint backlog details the work items that team members have agreed to complete in a given fixed-length sprint.
- **Sprint or Iteration** - a sprint is a fixed-length development period, or iteration, with a clear goal, consisting of an agreed set of work items to complete
- Each successive sprint builds on the last, and planning occurs between the sprints.
- **Daily Scrum** - 10 to 15-minute daily meetings
- “All work performed in Scrum needs a set of values as the foundation for the team's processes and interactions. And by embracing these five values, the team makes them even more instrumental to its health and success”. – Scrum Alliance
- Scrum values - Focus, Courage, Openness, Commitment, Respect.

XP (Extreme Programming)

- Key elements of XP:
 - Programmer Centric Model
 - Rapid Delivery
 - Small releases of software in 30 to 180 days cycles
 - Releases consist of iterations lasting 1 to 4 weeks
 - Small, self-directed team of programmers who own coding standards
 - Coding tasks based on a set of **customer user stories** – each of which **describes a single functionality** the customer requires of the final product
 - The focus of design is on simplicity, and ongoing testing, *refactoring* and customer feedback are integrated in the development process. **Refactoring refers to the re-design of a program or of code that has already been implemented to improve its performance.**
 - Core Principles – Pair Programming, Sustainable Pace, Ongoing Automated Testing



Lean

- Core principles derived from manufacturing companies such as Toyota
- Doesn't prescribe the use of specific development methods. Instead, it provides guidelines for **streamlining the development process** so that teams can meet customer needs much more efficiently
- Seven key principles:
 - Eliminate waste
 - Incorporate continuous learning
 - Delay decisions
 - Deliver software quickly
 - Empower the programming team
 - Focus on system integrity
 - Focus on the whole system



Kanban

- Kanban or “Card you can see” (in Japanese) was developed as a subcomponent of the Toyota Production System and has its origins in these Lean and Just In Time (JIT) manufacturing processes.
- In Kanban the workflow is visualized: work is broken down into small, discrete items and written on a card which is stuck to a board; the board has different columns and as the work progresses through different stages (e.g. ready, in progress, ready for review etc.) the card is moved accordingly.
- Like Scrum, Kanban encourages work to be broken down into manageable chunks and uses a Kanban Board (very similar to the Scrum Board) to visualize that work as it progresses through the work flow.
- **In Kanban, there are no required time boxes or iterations.** While the Kanban method is iterative in nature, the continual On scrum teams, there are at least three roles that must be assigned in order to effectively process the work: the Product Owner, Scrum Master, and Team Members. Each role has its own set of responsibilities, and they must work together to achieve an orderly and efficient balance.

Scrum and Kanban Boards – Similarities & Differences

- While very similar, the Scrum and Kanban Boards have significant differences.
- In a Scrum board, the columns are labeled to reflect periods in the work flow beginning with the sprint backlog and ending with whatever fulfills the team's definition of done. All the stories added to the board at the beginning of each sprint should be found in the final column at the end of that sprint or the sprint was unsuccessful. After the sprint retrospective, the board is cleared and prepped for the next sprint.
- On the Kanban board, **the team publishes the maximum number of stories allowed in each column at any one time**. This limit to Work In Progress rule in Kanban makes it suitable for teams with limited resources or where input from every member is required on each item. It will continue to flow for as long as the project continues, with new stories being added as the need arises, and completed stories being re-evaluated should it be necessary.

Unified Process

The Unified Process, or UP, is a detailed framework for the iterative and incremental development of software. Several agile methodologies have been derived from UP, including Agile Unified Process, or AUP, Essential Unified Process, or EssUP, and Open Unified Process, or OpenUP.

- AUP is a simplified version of the Unified Process, or UP. It adapts UP specifically for agile development. AUP projects proceed according to four phases – inception, elaboration, construction, and transition. Each of the four phases is divided into one or more iterations. As in other agile methodologies, the focus of each iteration is on developing a small, production-worthy piece of software.
- EssUP is a set of software development practices derived from UP, agile principles, and an approach to improving business processes known as Capability Maturity Model Integration. EssUP emphasizes the concept of **Separation of Concerns, or SOC**. SOC is aspect-oriented thinking, in which specific concerns (set of information that affects the code) are identified and then addressed in order of priority. Once you've identified and prioritized specific concerns, you can choose the EssUP practices that best address these concerns and incorporate them in your own development process.
- OpenUP is a development process that provides only fundamental content and guidance to project teams. OpenUP **uses scenarios, use cases, risk management, and an architecture approach** to support development. It focuses on collaborative project teams and can be used as is, or built on to suit different types of project.




Crystal

- Crystal Methods is a collection of Agile software development approaches, **focuses primarily on people and the interaction among them while they work on a software development project**
- Adopts colors to identify projects
 - Typically Crystal Clear is suitable for projects with small teams of up to six members - each iteration of between 60 to 90 days
 - Crystal Yellow is suitable for teams of seven to 20 -
 - Crystal Orange is for teams of 21 to 40 - expected to release working software every 90 to 120 days
 - Crystal Red is appropriate for teams of 41 to 80 members.
 - Crystal Maroon is for very large projects, with 81 to 200 or more team members
- Crystal uses a very adaptable approach to software development
- Key Practices -
 - An iterative and incremental development approach
 - Active user involvement
 - Delivering on commitments



Feature Driven Development (FDD)

- Provides a prescriptive model for development
- **All aspects of the software development process are planned, managed, and tracked at the level of individual software features.**
- FDD teams use short fixed-length iterations of two weeks or less, during which they work on a small set of assigned features. The team then moves onto a new set of features.
- FDD specifies five processes for teams to complete in order:
 - develop an overall model
 - build a features list
 - plan by feature
 - design by feature, and
 - build by feature



Dynamic Systems Development Method (DSDM)

- Structured framework for agile system development
- Unlike other agile methodologies, **it reflects a business perspective rather than a technical one.**
- One of its main goals is ensuring the fitness of developed products for their intended business purposes.
- A DSDM project is roughly divided into three phases –
 - activities that must occur before a project begins
 - those that make up the actual project life cycle
 - those that must occur once a project completes
- The project life cycle is further divided into five stages –
 - Feasibility Study
 - Business Study
 - Functional Model Iteration
 - Design and Build Iteration
 - Implementation
- Within each phase, DSDM defines structured sets of activities and the relationships among these.

Adaptive Software Development

- The ASD methodology views the development process as a collaborative learning cycle, with three key phases –
 - **Adaptive cycle planning or speculating** – During adaptive cycle planning, you need to plan the number and length of required cycles per development period, the objective statement for each cycle, and the technology and components needed.
 - **Collaborate** - Concurrent component engineering is the phase during which the team collaborates to implement the planned work. Team members or sub teams generally work on components simultaneously and then integrate their work products.
 - **Learn** - Each development cycle ends with learning, obtained through quality reviews. It's recommended that each quality review include feedback from end users who've tried using the developed software.

Test Driven Development

- TDD is a technique that **focuses on the development of unit tests for a particular code block before the code itself is written**. It originally derives from the test-first principles of XP.
- TDD is an effective approach when a team is already aware of the sort of code that will be used and how it will be designed.
- As code is written, it's checked with an associated unit test to ensure that it's functional. So **testing is an integrated part of the development process, rather than an activity that occurs only after code has been developed**.
- Used alongside agile methodologies, TDD provides useful guidelines for testing and documenting progress. Programmers can use test results to guide subsequent code development.



Agile Values and Principles

Primary and secondary Agile values

The agile approach represents a shift away from traditional project management. However, it doesn't dismiss traditional values outright. Instead the Agile Manifesto identifies primary values, which are agile, and secondary values, which are more traditional. Primary values are considered more important, but the secondary values are important too.

The Agile Manifesto rests on **four primary values**, each associated with secondary values:

- **individuals and interactions** over processes and tools
- **working software** over comprehensive documentation
- **customer collaboration** over contract negotiation, and
- **responding to change** over following a plan

Agile Principles – First Six

Twelve agile principles describe the four primary agile values in more detail. The first six principles are to:

- focus on satisfying the customer
- welcoming change
- delivering working software frequently
- ensuring that business people and developers work together
- motivating the individuals involved in development, and
- using face-to-face communication whenever possible



Agile Principles – Next Six

The remaining six agile principles are to:

- use working software as the primary measure of progress
- sustain a constant pace of development
- pay continuous attention to technical excellence and good design
- aim for simplicity
- use self-organizing teams, and
- use regular reflection on how to become more effective

Question – Agile Principles and Actions

You're working on an agile team that's developing a software product. Which actions are in keeping with agile principles?

- **Are these activities correct for Agile Principle or incorrect?**
 - Problems during development are automatically escalated to a team lead –
 - *Incorrect - collaborate and resolve problems themselves.*
 - The team ensures that the application supports as many functions as possible, given its time constraints –
 - *Incorrect - focus only on what the customer requires.*
 - The team releases a new functional feature every 2-3 weeks –
 - *Correct - deliver working software frequently.*
 - Team members are able to talk to the business stakeholder when they need more information on the feature they are building –
 - *Correct - business people and developers work together, preferably on a daily basis.*
 - The team tracks its progress based on how many fully functional software features they have delivered to the customer –
 - *Correct – measure progress based on working software.*

The Agile Project Management Model

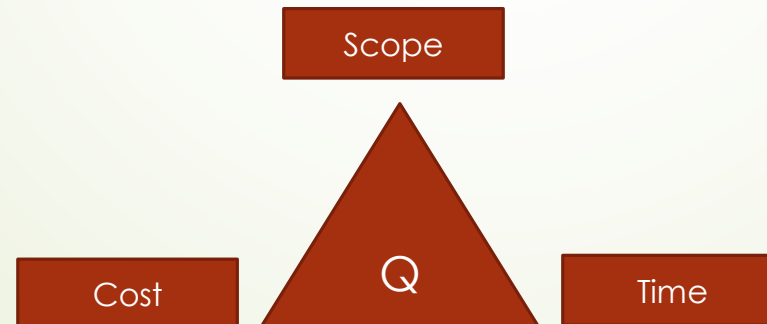


Role of the Project Plan (Group Discussion)

- Traditionally, a project plan is a document that helps project managers execute and control the phases of a project. It clarifies a project's objectives and how they can be achieved. Information included in a project plan typically includes the project's scope, cost, and schedule, as well as its activities, deliverables, milestones, and resources.
- Defined and empirical development models are associated with different attitudes toward the role of the project plan. A team using a highly defined approach like the waterfall model tries to keep as close to the project plan as possible, whereas an agile team using an empirical model regards the project plan as more flexible.

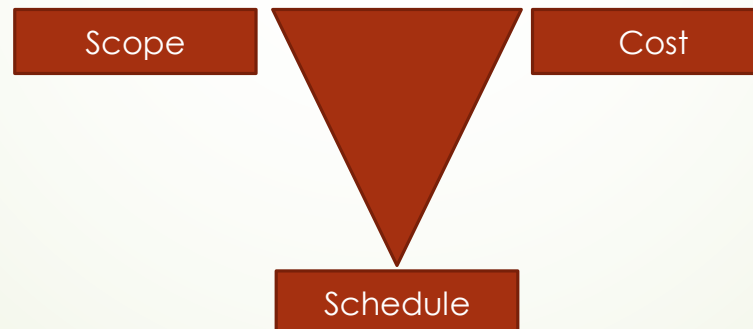
The Traditional Triangle of Constraints (Group Discussion)

- The traditional iron triangle of constraints identifies three main types of constraints on the success of a project – scope, cost, and schedule. Change to any one of these constraints will affect the others. The quality of a project depends on satisfying all three constraints. *The top point of a triangle is labeled Scope, and its bottom points are labeled Cost and Schedule. The center is labeled Quality.*
- **Scope** – Scope is the primary constraint because it's the first constraint that's known. Projects must meet the agreed scope – no more, no less.
- **Cost** - The constraint of cost refers to the limits posed by a project's budget. Projects must be delivered within cost.
- **Schedule** - The schedule constraint refers to the time limits posed by project deadlines, including interim milestones and a final release date. Projects must be delivered on time.



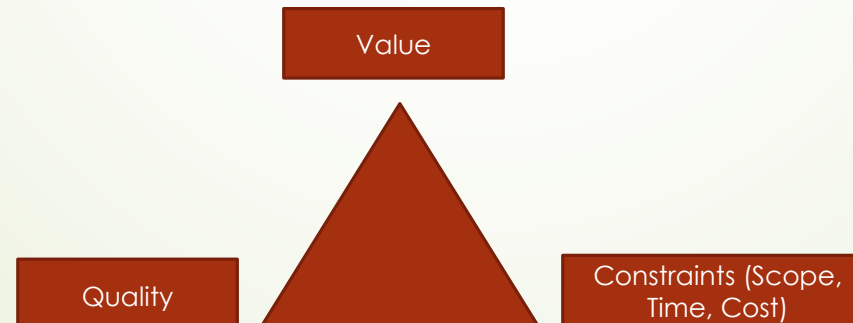
The Agile Iron Triangle of Constraints

- The agile approach rejects the traditional iron triangle because the traditional version measures success in terms of how well a project adapts to defined plans, instead of in terms of value to customers and the development of working software.
- What's known as the **Agile Iron Triangle** reverses the traditional triangle, recognizing the importance of meeting a fixed, or time-boxed, **schedule as the main constraint**. Cost and scope are the variables.
- *The agile iron triangle is upside down. The two points at the top are labeled Cost and Scope, and the bottom point is labeled Schedule.*
- Using this triangle, the success of a project is still measured in terms of how well a project meets a pre-determined scope, cost, and schedule. **As such, it still fails to recognize the value of a project's ability to change.**



The Agile Triangle of Constraints

- ▶ The agile triangle measures a project's success in terms of its ability to respond to change, or meet its main goal of delivering value to the customer. Other goals are to deliver quality while working within certain constraints.
- ▶ *In the agile triangle, the top point is labeled Value, and the bottom points are labeled Quality and Constraints.*
 - ▶ **Value** - Value is the main goal because the success of a project is measured by how much value it brings to the customer. The better you're able to adapt to and meet changing customer requirements, the more value you bring to the customer. To achieve this goal, it's possible to adjust quality and various constraints.
 - ▶ **Quality** - In an agile approach, you achieve the goal of quality by continuously delivering value to customers in the form of workable products.
 - ▶ **Constraints** - The constraints of a project are its scope, cost, and schedule. They're important because they can help you add to the value of a project, although they don't identify the project's actual goal. You adjust the constraints so you can meet the main goal of delivering value and quality to a customer.




Traditional and Agile Approaches Differences

- Agile project management differs from traditional project management in four fundamental ways: **high-level project scope, projects progress through multiple iterations, teams are self-organizing, and extensive customer involvement is required**
 - In traditional project management, the scope of a project – including the work it will include and the deliverables it will provide – is fully defined before project work starts. In contrast, the **scope details of an agile project continues to be adjusted** as the project evolves.
 - At the start of an agile project, the project team **defines a basic plan for each iteration**, or development cycle. This plan is general rather than detailed, **to allow for later modifications**. At the end of each iteration, the team reviews its progress and involves the product owner or customer to make decisions on what direction the next iteration should take.
 - Traditional project management relies on managing team members to ensure that the right work is done on time. In contrast, **agile teams are self-organizing and self-regulating**. Agile team members are expected to take responsibility for their own work, and for collaborating with one another. All team members should share accountability for the work completed.
 - Ongoing customer involvement is central to agile project management. **Customers help the project team define requirements and priorities throughout the project life cycle, and review the results of each iteration**. In traditional project management, there's typically little customer involvement once the development phase starts, until after project work has completed. Final deliverables are then handed over for customer review.

Agile Approach Phases

Based on the Agile Project Management model derived by Jim Highsmith, Agile Project Management can be divided into five phases – Envisioning, Speculating, Exploring, Adapting, and Closing.

- **Envisioning** - In the envisioning phase, you focus on determining and refining a project's vision. You also have to determine your project's scope – or requirements – develop a project schedule, and decide on your project team. If a project has tight time constraints, for example, you'll need to focus on choosing highly productive team members.
- **Speculating** - During the speculating phase, you develop estimated iteration and release plans and feature breakdown. Along with the project requirements defined in the envisioning phase, you can then develop a rough project plan. During the speculating phase, you also need to consider project risk and appropriate risk mitigation strategies, and estimate project costs.
- **Exploring** - During the exploring phase, the team designs, builds, and tests features in useful increments to the customer. Project leaders help developers overcome any obstacles that arise and manage the team's interaction with product owners, customers, or stakeholders.
- **Adapting** - In the adapting phase, the team takes feedback to review the results of an iteration and, based on the results, adapts product features and project plans for the next iteration. A project leader may also make adjustments at this point. For example, if any problems are raised by members of a project team or members aren't working well together, the project leader may make necessary changes.
- **Closing** - During the closing phase, all project work is completed. Lessons learned are recorded for future use and tasks associated with a final product release – like creating user documentation – may be completed.



Traditional and Agile Approaches Comparisons – Envisioning – Initiating

- **Envisioning - Initiating** - The Initiating phase of traditional project management involves defining the business objectives for a project and incorporating these in a clear vision statement. It also involves creating a detailed description of the project's scope and obtaining a project sponsor's approval to go ahead with the project.
- The envisioning phase is similar in that it involves determining the vision for a project and relevant objectives and constraints. However, it **doesn't involve defining a fixed project scope**. Instead the scope details of a project are expected to become clear only as the project progresses.



Traditional and Agile Approaches Comparisons – Speculating - Planning

- **Speculating - Planning** - During a traditional project's Planning phase, every aspect of the project is planned.
- The agile speculating phase also involves planning. During this phase, the team creates a **project or release plan based on required product features** and the project team's capacity. However, this is a rough estimate and it's assumed that this plan will be adjusted as work proceeds.



Traditional and Agile Approaches Comparisons – Exploring - Executing

- **Exploring - Executing** - During the traditional Executing phase, teams carry out project work in accordance with the specifications in the project plan. Project managers track the actual work completed against the work that's been planned.
- The agile exploring phase **also involves completing actual project work**. However, it's not expected that a development team will adhere strictly to the initial project plan. Instead the **process includes ongoing reviews and testing, with adjustments made to the plans for subsequent iterations** based on the results.



Traditional and Agile Approaches Comparisons

– Adapting - Monitoring & Controlling

- **Adapting - Monitoring & Controlling** - The traditional Monitoring & Controlling phase occurs throughout a project. It involves comparing actual and original planned results, and taking steps as necessary to bring these in line.
- In the agile adapting phase, the focus is on **adapting to and incorporating change** in order to develop the most value for the customer within acceptable constraints.



Traditional and Agile Approaches Comparisons – Closing

- **Closing** - In a traditional project, the Closing phase involves completing relevant documentation and handing deliverables over to the customer, or project sponsor.
- In an agile project, the closing phase begins only after several iterations, each of which creates one or more deliverables that are handed to the customer.



Agile Project Management – Roles – Project Leader

- In the agile approach, teams are usually self-organizing. Team members understand how they can contribute to a project and regulate their own work. This means there's **less room for the explicit leadership roles** you find in traditional projects.
- **Agile project managers are often referred to as project leaders.** They take over a lot of the traditional project manager's responsibilities. Project leaders in the Scrum projects are called 'Scrum Masters' and some organizations use the traditional Business Analyst to assume this role for the Agile projects. In agile methodologies, project leaders may also take on more of a coaching role, because instead of delegating tasks and overseeing team members' work, they focus on helping teams achieve their goals.

Agile Project Management – Roles – Business Analyst (Group Discussion)

- In a traditional development team, the business analyst's role is to be liaison between the customer (internal or external) and the development team. In an agile project team, the role of the business analyst varies depending on the organization and the complexity of the project. Most companies which have a mixed (traditional & agile) portfolio will have the BA positioned along side the Product owner or assume the role of the Product Owner or report to the Product Owner.
- In Agile, the BA will manage constraints or needs the team has, and to help team members express themselves in business terms that the customer can understand. For example, the business analyst ensures that new product functionality is explained to the customer. If a development task is taking longer than anticipated to complete, it's also the business analyst's responsibility to ensure that the customer understands why and work alongside the Scrum Master and Product owner to make necessary adjustments to sprint and product backlogs.



Agile Project Management – Roles – Scrum Master

- In agile methodologies such as Scrum, a Scrum Master facilitates the work of project team members and achieve project goals. In agile projects, progress of the project is led by the Scrum Master, who is responsible for tracking and monitoring progress using tools such as a burndown chart and Scrum task boards.
- If problems such as delays or conflicts arise, the Scrum Master is expected to help resolve them.
- If the need arises, Scrum Masters also take on more traditional project management roles. For example, they may need to provide leadership if a team isn't succeeding in regulating itself. They're also expected to track and manage the progress of projects and hold regular status, or stand-up, meetings.
- Larger projects can have multiple Scrum Masters.



Agile Project Management – Roles – Product Owner/Product Manager

- An agile product manager, or product owner, is responsible for ensuring that the product delivered through a project meets customer needs. This role can be a person from the business.
- The Product Manager should manage and evangelize the vision of the product, gathering stakeholder feedback for use in generating user stories, and prioritizing release planning.
- If stakeholders are unhappy with a product feature, for example, the product owner/manager needs to communicate with the project team so that the problem can be rectified.
- A product owner/manager should:
 - Understand the market and product positioning.
 - Perform feature analysis – what stays, what gets pushed out and what goes away.
 - Manage time efficiently so that budget restraints can be met, and
 - Work full-time on each project iteration.

Question - agile team roles to their responsibilities

- Who does this:
 - Facilitates the work of project teams
 - **Project Leaders or Scrum Masters** - *Project leaders in agile projects are often called coaches. They facilitate the work of self-organized teams, rather than delegating and overseeing work.*
 - Supports an on-site customer and refines the customer's requirements
 - **Business Analyst** - *In an agile project, a business analyst is responsible for supporting an on-site customer, and for working alongside other team members to clarify the customer's needs.*
 - Helps team members achieve project goals by holding daily stand-up meetings
 - **Scrum Master** - *The Scrum Master helps team members achieve the goals for each iteration, and ensures that they keep a project moving in the right direction.*
 - Promotes the vision for a product and prioritizes project requirements
 - **Product Manager** - *Product managers maintain and promote the vision for the product that a project will deliver. Often this role is filled by an on-site customer.*



Agile Project Management – Team Roles

- **Designer** - In agile methodologies such as Extreme Programming, or XP, team members include one or more designers. They work alongside product developers to help simplify complex designs.
- **Developer** - Agile team members may have the role of developer. Developers are involved in building the product or service. In the case of a software project, the team may be composed of a number of programmers, that work together to build and test the product code.
- **Testers** - Testers help investigate products for possible flaws, and test product performance, scalability, and stability in relation to customer requirements. To do this effectively, they must be able to envision all possible uses of a product. In agile software development projects, testing occurs on an ongoing basis.



Summary – Agile Project Management

- Agile project management can be divided into five phases – envisioning, speculating, exploring, adapting, and closing.
- Key characteristics of the agile approach are that the development process involves multiple iterations, project scope is defined at a high level only at the outset, agile teams are self-organizing, and extensive customer involvement is required throughout the development process.
- In agile projects, key roles include project leaders, business analysts, Scrum Masters, and product managers. In addition, team members fill roles as designers, developers, and testers.
- Agile development avoids the prescriptive, plan-oriented approach associated with traditional project management, and makes use of self-organizing teams. However, it's a common misconception that agile projects don't require project management. Project management is still necessary. However, the traditional responsibilities of the project manager may be handled differently and possibly be spread out across members of the agile project team.



Deep Dive into Important Elements of Scrum

Defining Scrum

- The term "scrum" originates from the rugby formation, in which a team's players work together to gain possession of the ball.
- Scrum enables project teams to develop complex products quickly and efficiently, to adapt to change, and to regularly deliver value to customers in the form of working products.
- Scrum aligns to core agile values in several ways:
 - it involves an iterative, incremental development process
 - it focuses on frequent delivery of working product features to the customer
 - it depends on a high level of customer involvement throughout the development process, and
 - it relies on self-organizing and cross-functional development teams
- Many Scrum practices are common to all agile methodologies, although associated terminology may differ. For example, most agile methodologies divide the development process into short, incremental periods of development. In Extreme Programming, or XP, and general agile terms, these are known as iterations. In Scrum, they're called sprints.
- Like other agile methodologies, Scrum is based on empirical, rather than prescriptive, process control.
- Scrum projects can proceed with even a high-level product backlog. Instead of defining every aspect of a product at the start of the project, a **Scrum team follows a continuous cycle of inspection and adaptation.**
 - Inspection - Inspection occurs regularly during the development cycle to uncover problems and deviations, and enable customers to provide feedback and request changes. Examples of inspection include events such as sprint planning, the daily scrum, the sprint review, and sprint retrospective.
 - Adaptation - Adaptation occurs in response to the results of inspection. Change is implemented as soon as possible after the need for a change is identified. This helps prevent issues from developing into bigger problems and becoming more difficult to resolve.

Scrum Process – Pre Game Phase

- The core of Scrum, originally referred to as the "game" by its creators, describes how to prepare and run Sprints. While not officially described as such in the Scrum guide, the phases of a Scrum project cycle could be considered and are sometimes described as pre-game, game, and post-game.
- In the first phase, the product owner defines the project's goal and develops the initial product backlog. The product backlog will continue to be updated as additional or modified requirements emerge during the project. Scrum development work is done during the actual development or game phase, where the product is built, inspected, and adapted using an iterative process. Each sprint produces a potentially releasable product which the customer may or may not decide to release at the completion of a sprint phase.
- The pre-game phase includes two activities – planning and the development of the product's high-level design.
 - **During planning, the product owner, with the help of the customer, identifies the most important requirements of the product that's to be developed.** The decisions are based on what the market demands and on what customers believe will provide the most value for their organizations. The product owner captures these requirements in the project's goal and the product backlog.
 - **During development, the product owner and the customer develop a list of prioritized requirements, known as the product backlog.** Developers will determine what functionality they need to develop to meet these requirements. The product owner maintains the product backlog throughout the course of the project and updates it as requirements change or are updated.



Scrum Process – Game Phase (Group Discussion)

- The game phase refers to the sprint, or development, phase.
- Sprint Planning takes place at the start of a sprint,
 - **Formulates a sprint goal**
 - Re-Prioritize backlog items
 - Divide each backlog item into tasks
 - Estimates the effort it should take to complete each task
 - Design and develop product backlog items
 - Backlog grooming
 - Daily Standups

Scrum Process – Post Game Phase

- The work needed after a sprint or series of sprints to release the product, is sometimes referred to as the post-game phase.
- The goal of each sprint is to produce a releasable product increment. Depending on the nature of the product, the customer may release the latest developed functionality after each sprint or decide not to do so.
- In some cases, the product may be released at specific release intervals after a series of sprints, and in other cases the entire product backlog needs to be delivered before the final product is released.
- **Scrum development demands that at the end of each and every sprint, a functioning product can be handed over to the customer. This means that when Scrum is implemented correctly, there is no post-game phase. This is because every sprint delivers production-ready product and thus no additional work is needed to release the work.**
- If a product has multiple releases, the project is not yet in closure and work continues towards the next release. In some of these projects, closure tasks, such as creating documentation or integration testing, may be listed as a backlog item to be completed as part of the sprint goal.

Deep dive into scoping for agile projects and requirements management using agile tools



Agile Project Planning Activities

- Agile project planning is cyclical and ongoing, with different types of planning repeated throughout the project life cycle.
- **Project planning is usually either date-driven or feature-driven.**
 - In a **date-driven - or time-boxed - project**, the release date is set but the set of features that will be included in the product release is negotiable.
 - In a **feature-driven - or scope-boxed** - project, the set of features to be developed is agreed but the release date isn't set. Instead a product will be released once all features have been developed.
- The approach that's chosen will depend on the customer's business needs.
- In most cases, date-driven planning is preferable because it forces customers to balance the importance of features against the time it will take to develop them. This tends to result in better overall value because it helps prevent spending on unnecessary or unimportant features.
- However, it is possible to use scope-driven planning with an agile approach. A set of required features is selected but can be adjusted as the project progresses and more information is gained through feedback. This approach is appropriate for highly experimental projects, in which it's difficult to estimate how long work will take to complete.
- It's also possible to take a hybrid approach to planning a project. For example, a manager who's setting out to manage a feature-driven project might discover that certain key team members will be unavailable after a certain date.

Question

- As a Product Owner, you discover that it's of vital importance for the company to release the application before any of its competitors can release similar applications. Which planning approach should you use – Feature-driven or Date-driven?
 - **Feature Driven:** *Incorrect. In a feature-driven project, the completion of a set of features is prioritized above meeting a particular release date. In this case, however, an early release date is vital.*
 - **Date Driven:** *Correct. In a date-driven project, the completion date is fixed and the set of required features is negotiable, which is ideal in this case because of the urgency involved in getting the application to the market.*

Release Planning Steps (Group Discussion)

- The first stage of agile planning is release planning, which provides high-level information about how much work can be completed by a particular date. The key tasks involved in release planning include establishing project goals, creating user stories, prioritizing stories, estimating stories, grouping stories, and setting a release date.
 - Establishing Project Goals –
 - Gathering Requirements -
 - Creating User Stories -
 - Prioritizing Stories –
 - Estimating Stories –
 - Grouping Stories –
 - Setting a Release Date –
- Once a team has created a release plan, the plan should be kept in an accessible place so it can be referenced by the team and stakeholders. It will also be updated as the project progresses and requirements change.

Iteration/Sprint Planning (Group Discussion)

- Similar to the release plan, an iteration plan can be documented in various ways – for example, using a spreadsheet or a board and note cards. The most important thing is that it's easy to see which tasks have been assigned to an iteration. During iteration planning, an agile team plans what to include in the upcoming iteration in more detail. The core tasks involved in iteration planning include updating requirements, confirming user stories and priorities, decomposing selected stories into tasks, and refining estimates.
 - Updating Requirements –
 - Confirming User Stories and priorities -
 - Decomposing selected stories into tasks -
- Refining Estimates - Release and iteration planning are similar in that both involve estimating work, prioritizing and selecting stories for development, and setting a schedule. Also, like iteration planning, release planning activities may be repeated multiple times as the team progresses through the project.

Estimation Tool - User Story Points

- When estimating the relative sizes of user stories, agile teams often use *story points*. Each story point represents a fixed amount of development effort, or work.
- For example, a fairly small coding task might be assigned five story points. If another task is judged to require three times as much work, it will be assigned 15 story points. On their own, story points don't tell you how long each user story will take to develop. **Story points should not be seen as equivalent to hours and the actual story point value assigned could vary greatly from one team to the next. The key is to use them as ratings of size in relation to other stories in the backlog.**
- It's important not to confuse the **size of a task** – which is measured in story points – **with its complexity**. Although size can sometimes be an indicator of complexity, it's possible that a relatively simple story will involve as much work, and therefore be allocated the same number of points, as a complex one.
- For example, the task of capturing data in order to populate a database is simple but, depending on the amount of data that's involved, might involve as much or more work than a far more complicated task, like designing a system's user interface.

Estimation Tools – Planning Poker

- A popular agile estimation technique is known as *planning poker*. Planning poker involves using a pack of agile planning cards, arranged in a deck, to represent possible numbers of story points for assigning to user stories. It's an adaptation of a technique called wideband Delphi, in which a group of people choose estimates independently, compare and discuss their individual results, and then revise their estimates to arrive at a consensus.
- In planning poker, each card in the deck is marked with a different number of story points, and together the pack represents a rating scale. This scale isn't always even, or linear, though.
- It's useful to use small intervals in numbers of story points only when comparing small stories – such as if one story is valued at 2 points and another, involving twice as much effort, at 4 points.
- The cards in the planning poker deck are each labeled with a unique value. The most common values in the series are a question mark, 0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, and 100, as well as a card labeled with the infinity symbol. Each value represents a number of story points
- For larger stories, it's better to use even larger intervals. For instance, it's unlikely to be useful to assign one story 20 points and another 21 or 22 points because the values are so similar. Instead it would make sense to assign both stories the same number of points. You might then assign a story that clearly requires much more effort 40 points.

Estimation Tools – Fibonacci series

- The Fibonacci series is a famous series of numbers in which each number is the sum of the previous two numbers. This results in increasingly larger intervals between numbers as the series progresses, such as the numbers 1, 2, 3, 5, 8, 13, 21 and so forth.
- The sequence used to mark the cards in planning poker is typically a variation on the Fibonacci series in which the value 20 used in planning poker would typically be a value of 21 on the Fibonacci scale.
- Although it's popular to use a scale based on the Fibonacci series, you can also choose to use either linear or binary scales.

Kano Model

- ▶ **The Kano Model helps product team understand and categorize 5 types of Customer Requirements (or potential features) for new products and services.**
- ▶ It was created in the early 80's by Japan's professor Noriaki Kano. The main purpose of the Kano Model is:
 - ▶ To communicate 5 universal categories of customer requirements that all product and service developers need to be aware of in order to remain competitive.
 - ▶ To show how each of these 5 universal categories can influence satisfaction and dissatisfaction.
 - ▶ To show how 2 of the categories add value and 2 of the categories detract from value, and 1 of the categories creates new value.
 - ▶ To help organizations understand their customer needs better than their customers understand their own needs.
 - ▶ To provide a mechanism to help organizations understand and classify all potential customer requirements or features into these 5 categories so they can prioritize development efforts on the things that most influence satisfaction and loyalty. This is done by the Kano Survey, or sometimes called a Kano Analysis.
 - ▶ Source – www.kanomodel.com

Kano Model Categories

- **One-dimensional Quality** - These attributes result in satisfaction when fulfilled and dissatisfaction when not fulfilled. These are attributes that are spoken and the ones in which companies compete. An example of this would be a milk package that is said to have ten percent more milk for the same price will result in customer satisfaction, but if it only contains six percent then the customer will feel misled and it will lead to dissatisfaction.
- **Must-be Quality** - Simply stated, these are the requirements that the customers expect and are taken for granted. When done well, customers are just neutral, but when done poorly, customers are very dissatisfied. Kano originally called these "Must-be's" because they are the requirements that must be included and are the price of entry into a market.
- **Attractive Quality** - These attributes provide satisfaction when achieved fully, but do not cause dissatisfaction when not fulfilled. These are attributes that are not normally expected, for example, a thermometer on a package of milk showing the temperature of the milk. Since these types of attributes of quality unexpectedly delight customers, they are often unspoken.
- **Indifferent Quality** - These attributes refer to aspects that are neither good nor bad, and they do not result in either customer satisfaction or customer dissatisfaction. For example, thickness of the wax coating on a milk carton. This might be key to the design and manufacturing of the carton, but consumers are not even aware of the distinction.
- **Reverse Quality** - These attributes refer to a high degree of achievement resulting in dissatisfaction and to the fact that not all customers are alike. For example, some customers prefer high-tech products, while others prefer the basic model of a product and will be dissatisfied if a product has too many extra features.[1]
- Source – www.kanomodel.com

How to be a leader on agile projects





Project Leadership – Differences between traditional vs. Agile

- ▶ **Decision Making** - In traditional project management, the project manager is responsible for the majority of decisions, and for solving problems that arise. The manager determines development strategies and then directs developers on how to implement these.
 - ▶ In agile project management, all members of the development team are responsible for discussing problems and coming up with solutions.
- ▶ **Project Planning** - In traditional project management, the project manager is responsible for creating a project plan, which outlines a project's goals and the activities it will involve, in virtual isolation.
 - ▶ In agile project management, all members of the project team collaborate to develop project plans.
- ▶ **Management** - The focus of traditional project management is on control. The project manager's main responsibilities are ensuring that everything proceeds according to plan, and directing the activities of team members.
 - ▶ Agile project management, however, is adaptive and flexible. Plans are periodically adjusted and teams are self-managing, and the project manager serves mostly as a facilitator and guides the development process based on Agile values and principles.
- ▶ **Reporting and tracking** - In traditional project management, reporting and tracking activities are formalized. Usually the project manager records the progress of a project in a log.
 - ▶ In agile project management, however, these activities are less formal. They're built into the agile process, through the use of project backlogs and burndown charts.

Servant Leadership

➤ **Safeguard the Development Process –**

- Safeguard the development process by facilitating meetings,
- Keeping the team from being distracted,
- Removing roadblocks that may hinder progress.
- Helping team members to communicate.

➤ **Communicate Progress and Issues –**

- Team members report problems to project leaders, who then follow up on them so the team can concentrate on its work.
- Responsible for mediating disputes within the team
- Communicating with external stakeholders on behalf of the team.
- Communicate the project goals and progress using daily stand-ups, task boards and burndown charts, which should be kept where all team members can see them.

➤ **Build a development community –**

- Build a development community by sharing personal experiences with the other team members.
- Create a safe development environment that allows for effective collaboration and experimentation.
- Act as an interface between the team and the organization.



Measuring success of agile projects

Product Quality – Traditional vs. Agile

- The success, or quality, of a traditionally managed project is defined in terms of how well the project meets time, budget, and scope requirements.
 - In an agile project, however, quality is judged in relation to meeting a customer's needs – and it's recognized that these needs may change over time.
- Quality assurance in a traditional project involves conforming to specifications.
 - In an agile project it involves ensuring a product's fitness for its intended business use. Agile teams need to focus on both customer-defined quality and technical quality. Customer-defined quality refers to what the customer will accept and be satisfied with. This depends on the customer's needs and preferences.

Burndown Charts

- A sprint burndown chart tracks the cumulative number of hours of work remaining in a sprint against the number of days left in the sprint. It typically includes a line indicating an ideal progression – which involves steadily decreasing the amount of remaining work over time – and another line indicating the team's actual progress.
- *An example burndown chart includes development hours, from zero to 400, on its y-axis and the number of days in a sprint on its x-axis.*
- A burndown chart makes it possible to check a team's progress at a glance. It's also a good visual indicator of the team's velocity - the rate at which work is being completed - compared to the estimated velocity.
- A team can use burndown charts to track progress at both the project and sprint levels. When this type of chart is used to track progress toward a final product release, it's known as a release, or project, burndown chart.

Burndown charts and Progress charts

- Both burndown charts and progress charts let you compare actual and estimated values, and both provide a quick, highly visual way to track progress.
- Burndown charts are designed to monitor work effort that's remaining. A sprint burndown chart tracks the cumulative number of hours of work remaining in a sprint against the number of days left in the sprint. It typically includes a line indicating an ideal progression – which involves steadily decreasing the amount of remaining work over time – and another line indicating the team's actual progress.
 - *An example burndown chart includes development hours, from zero to 400, on its y-axis and the number of days in a sprint on its x-axis.*
- Progress charts are intended to track the status of individual tasks – work that hasn't been started, that's still in progress, or already complete.
 - An example of a commonly used progress chart is a task board. In its simplest form, a task board makes it possible to see at a glance which tasks still have to be started, which are in progress, and which have been completed.
- Task boards like this are also referred to as "information radiators." In agile projects, information radiators are publicly displayed and used to convey status information to a team.



Q & A