## 1 - Introduction to Agile Methods

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, 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.

### 1.1: Traditional Software Development Practices

The **Traditional Software Development Practices** include

* Waterfall
* Spiral
* V-Model
* Rational Unified Process
* Rapid Application Development

These methods are the most commonly used approach by organizations where software development activities are completed sequentially.

### 1.2: Motivation for Agile Methods

Agile methods are being adopted by many firms to help improve team performance, customer happiness, and project adaptability. These methods enable organizations to adjust more quickly to market changes and complete more projects effectively. Agile training is a great approach to get your company and project team up to speed on the fundamentals of Agile.

### 1.3: Benefits of Agile Practices

End-user participation is encouraged throughout the project, ensuring visibility and transparency. There is continuous planning and feedback throughout the process, making sure that the project delivers value to the business from the start.

Companies are embracing the concept of generating business value early in the development process, making it easier to minimize development risks.

Some of the main benefits of agile project management are:

* High Product Quality
* Higher Customer Satisfaction
* Increased Project control
* Reduced Risks
* Faster ROI
* Benefits according to the Additional Notes  
    
   **Hands-on business outputs** involve regularly inspecting outputs firsthand to determine whether business requirements are being met – and whether the business value is being delivered for the organization.  
    
   **Direct stakeholder engagement**: involves actively engaging internal and external customers throughout a process to ensure that the resulting deliverables meet their expectations.  
    
   **Immovable deadlines**: are fixed time commitments that encourage staff members to deliver regular ongoing value to the organization.  
    
   **Management by self-motivation**: involves using the power of self-organized teams to deliver outcomes under the guidance and oversight of the customer.  
    
   ‘**Just-in-time’ communication**: replaces traditional corporate meetings with techniques for more effective communication and knowledge transfer (Differ Commitment)  
    
   **Immediate status tracking** provides tools that enable staff to keep others in the organization continuously aware of the status of the work that they are doing.  
    
   **Waste management: involves** maximizing the value of the organization’s resources by reducing and, where possible, eliminating low business-value activities.  
    
   **Constantly measurable quality**: involves creating active checkpoints where organizations can assess outputs against both qualitative and quantitative measurements.  
    
   **Rearview mirror checking** provides staff with tools for regularly monitoring and self-correcting their work.  
    
   **Continuous improvement**: involves regularly reviewing and adjusting business activities to ensure that the organization is continuing to meet market and stakeholder demand.
* Benefits according to Presentation  
  + Responsive planning
  + Business-value-driven work
  + Hands-on business outputs
  + Direct stakeholder engagement
  + Immovable deadlines
  + Management by self-motivation
  + Just-in-time’ communication
  + Immediate status tracking
  + Waste management
  + Constantly measurable quality
  + Continuous improvement

## 2 - Basics of Agile Software Development

* A cross-functional team of developers, testers, and analysts is assembled. If Scrum is being used, a Scrum Master is on point to act as the facilitator of the team, working closely with the Product Owner/client.
* A Sprint Planning meeting is organized to plan key activities and projects into a manageable development cycle of 2 to 4 weeks. To give engineers a clear path to success, requirements are broken down into tasks and hours.
* When it's time to start working on a project, all requirements are placed in the "product backlog" and then moved to the "sprint backlog" during "sprint planning." Due to stakeholder feedback and changing business requirements, the "product backlog" is constantly evolving and re-prioritized.
* **Daily Scrums** are held for a max of 15 minutes to review **“work in progress”** during an iteration. The focus for standups for each team member is (a) what did you accomplish yesterday (b) what is your focus for today (c) what obstacles are in your way?
* Daily Scrums are typically organized around a task board with columns and swim lanes. Many more columns can be added to the task board to break down the key stages of the job flow. It’s important to note that the task board can be customized so that it works for your team and your projects get creative. The primary goal of Agile is to stay open and adaptable rather than being "boxed in."
* Sprint Review and Sprint Retrospective meetings are held at the end of each sprint. The Sprint Review's main goal is to "show the world" what you've accomplished and got the Product Owner to accept completed user stories. The Sprint Retrospective is used to evaluate the agile process and adapt/improve it for the team.

### 2.1: Iterative & Incremental Approaches

Incremental development is a method of product development that splits the product into completely functional pieces called increments whereas Iterative development is when a team gradually adds features and functionalities but does not wait until each one is finished before releasing it.

### 2.2: Popular Agile Methods

The following is a quick overview of the many types of Agile Methodologies:

1. **Agile Scrum Methodology** Scrum is a lightweight agile project management methodology that can be used to manage iterative and incremental projects.  
     
    It has gained popularity over the years due to its unique qualities like simplicity, sustained productivity, and the ability to blend several underlying approaches adopted by other agile methods.
2. **Lean** It is the iterative, agile methodology that directs the team on addressing customer values by compelling value stream mapping, although, it is a deeply adaptable, emerging methodology with the absence of solid guidelines, laws, or methods.  
     
    In comparison to the hierarchical flow of control, Lean delegates decision-making authority to individuals and small groups. It focuses on the effective utilization of team resources and ensures that everyone is as productive as possible for maximum time.
3. **Kanban** Kanban is a highly visual workflow management approach popular among Lean teams that can be used to see and carefully maintain the making of products. It focuses on continuous product delivery without putting excessive stress on the entire software development life cycle.  
     
    Kanban, like Scrum, is a technique for more efficiently enabling collaborative teamwork. However, by articulating the best possible team workflow, Kanban encourages consistent cooperation and maintains active ongoing learning and enhancement.
4. **Extreme Programming(XP)** Generally used with Scrum, it can focus on how Agile can enhance customer happiness; instead of giving everything the client wants soon, it gives them what they want now. XP focuses on consistent propaganda and accurate development cycles. In addition, it employs code review, pair programming, and frequent customer interaction.  
     
    XP method is based on the \*\*\*\**four simple values:*
   * *Uniformity,*
   * *Simplicity*
   * *Communication,*
   * *Feedback and*
   * *Endurance*.
5. **Crystal** The Crystal Methodology is a group of smaller agile methods that include Crystal Clear, Crystal Yellow, Crystal Red, and many others. Every technique has its own set of guidelines. It was introduced by Alistair Cockburn while working on the Agile manifesto for software development.  
     
    The most number of individuals can be involved in a project depending on its scale; the larger the project, the more people participating. If the project is large, many roles can be accounted for and vice versa. Furthermore, interaction, people, expertise, communication, talents, and community are all converged in the crystal methodology.
6. **Dynamic System Development Method (DSDM)** DSDM gives a broad work-frame that is outlined;  
   * *To plan, handle, execute, and scale the complete process of software development,*
   * *It directs on the business-driven approach and*
   * *It doesn’t negotiate on quality and timely delivery.*
7. This is, however, an iterative, incremental, and Rapid Application Development (RAD) approach. Feasibility and business studies, prototype iteration and functional model, design and development iteration, and deployment are all part of the DSDM framework.
8. **Feature Driven Development (FDD)** FDD is a lightweight, agile software development approach that uses an iterative and accumulative development process. It incorporates a variety of industry-accepted best practices into a cohesive whole. These approaches are promoted from a feature-first viewpoint, to provide customer value.  
     
    However, the main purpose lies in delivering stable, working software frequently on time. FDD is a constructive, arranged, centered method for the entire software workflow management and a reliable choice for software-focused teams and associations

## 3 - Agile principles and manifesto

### 3.1: Vision and Principles of Agile Methods

The Agile Manifesto is a document that identifies four key values and 12 principles that its authors believe software developers should use to guide their work.

1. Customer satisfaction through continuous delivery of the product
2. Divide large chunks of work into smaller, more manageable jobs to speed up completion and make change integration easier.
3. Meet the agreed-upon timeline for delivering a functional product.
4. To guarantee that the project is moving in the right direction, all stakeholders must work together regularly.
5. Encourage team members to get the task done by creating a supportive environment.
6. Prefer face-to-face communication over other methods
7. The major indicator of progress is working software.
8. Try to maintain a constant pace of development.
9. Maintain quality of the product by paying attention to technical details.
10. Keep things simple.
11. Encourage team members to organize themselves.
12. Evaluate your performance regularly to improve.

## 4 - Agile Methodologies

### 4.1: Overview of SCRUM Methodology

Scrum is a framework for facilitating teamwork. Scrum encourages teams to learn from experiences, self-organize while working on an issue, and reflect on their successes and failures to continuously improve, much like a rugby team (from which it gets its name).

### Benefits of Scrum Methodology

Organizations that have adopted agile scrum have experienced:

* Higher productivity
* Better quality products
* Reduced time to market
* Improved stakeholder satisfaction
* Better team dynamics
* Happier employees

## Components of Agile Scrum Development

The scrum methodology is defined by team roles, events, artifacts, and rules.

## The Scrum Team

Scrum teams typically include 7 +/-2 members and no team leader to delegate tasks or decide how to solve a problem. The team as a whole decides how to deal with challenges and issues. Each member of the scrum team is an intrinsic element of the solution and is expected to carry a product from inception to delivery. In a scrum team, there are three main roles:

* **The product owner** The product owner is a crucial stakeholder in the project, usually an internal or external customer or a customer spokesman. There is just one product owner who conveys the team's overall mission and vision for the product. Lastly, the product owner is responsible for managing the product backlog and accepting completed work increments.
* **The scrum master** The scrum master serves as a servant leader to the product owner, development team, and organization. The scrum master ensures that the team follows Scrum theory, practices, and regulations. He or she has no hierarchical control over the team and acts rather as a facilitator. The scrum master protects the team by doing everything possible to ensure that it performs at its best.
* **The development team** The development team is a self-organizing, cross-functional team that has all of the necessary skills to deliver shippable increments at the end of each sprint. Scrum redefines "developer" to include everyone who contributes to the production of the delivered increment, rather than only programmers.

## Scrum events

* ***The sprint*** A sprint is a timed period in which specified work is finished and prepared for evaluation. Sprints can last anywhere from two to four weeks, but they can also be as brief as one week.
* **Spring Planning** Sprint planning team meetings are timed events that determine which items from the product backlog will be delivered and how the work will be completed.
* **Daily Stand-Up** The daily stand-up is a brief communication meeting (no more than 15 minutes) in which each team member rapidly and summarises progress since the last stand-up, scheduled work before the next meeting, and any impediments to their progress.
* **The Sprint Review** The sprint review is a "show-and-tell" or demonstration event in which the team shows off the work they did during the sprint. The product owner reviews the work and either approves or rejects it based on pre-defined acceptance criteria. Stakeholders or clients provide input to ensure that the increment delivered met the business requirement.
* **The Retrospective** The retrospective, also known as a retro, is the team's final meeting of the sprint to decide what went well, what didn't, and how the team can do better in the next sprint. The retrospective, which is addressed by the team and the scrum master, is an important opportunity for the team to reflect on its overall performance and identify strategies for continuous process improvement.

## Scrum Artifacts

* **Product Backlog** The product backlog is the single most essential document that lays out all of a system's, project's, or product's requirements. The product backlog may be viewed of as a to-do list of work items, each of which results in a business-valued deliverable. The product owner prioritises backlog items based on their business value.
* **Sprint Backlog** A sprint backlog is the specific list of items taken from the product backlog which are to be completed in a sprint.
* **Increment** An increment is the sum of all completed product backlog items since the last software release. While the product owner determines when an increment is delivered, the team must ensure that everything contained in the increment is ready to be released.

## Scrum Rules

The agile scrum rules should be determined by the team and based on what works best for their processes. The best agile coaches will tell teams to start with the basic scrum events outlined above, then inspect and adapt based on your team's specific needs so that collaboration improves over time.

## Practicing scrum

To get started with scrum, it is not uncommon for an individual scrum team to manage the product backlog and the progress of sprint backlog items in each sprint using simple scrum tools such as a whiteboard, sticky notes, or a spreadsheet. Scaling agile practices to the rest of the organization is unquestionably more challenging - the more scrum teams an organization has or the more geographically dispersed they are, the more cumbersome simple tools like whiteboards, sticky notes, and spreadsheets become.

### 4.2: Extreme Programming (XP)

One of the many Agile frameworks used by IT companies is Extreme Programming (XP). However, XP stands out from the other methods because of its focus on technical aspects of software development.

XP is a set of engineering practices. Developers have to go beyond their capabilities while performing these practices. That’s where the “extreme” in the framework’s title comes from.

## Phases in XP

The XP framework normally involves 5 phases or stages of the development process that iterate continuously:

1. The first stage is planning, which involves the customer meeting with the development team and presenting the requirements in the form of user stories that explain the desired outcome. After that, the team estimates the stories and prepares a release plan with iterations to cover the desired features piece by piece. If one or more of the stories cannot be estimated, so-called spikes can be used to indicate the need for more research.
2. Although designing is a part of the planning process, it can be distinguished to underline its importance. It's to do with simplicity and is one of the key XP values we'll go into later. A good design gives the system logic and structure while avoiding needless complexity and redundancies.
3. Coding is the phase in which actual code is written adopting XP practices like coding standards, pair programming, continuous integration, and collective code ownership.
4. Testing is the core of extreme programming. It is a regular activity that involves both unit tests and acceptance tests.
5. It's all about continual communication and feedback when it comes to listening. Customers and project managers are involved in describing the expected business logic and value.

## Roles in XP

Extreme programming emphasizes the significance and necessity of social skills such as communication, cooperation, responsiveness, and feedback by placing people at the core of the system. As a result, these roles are commonly related to XP:

1. Customers are expected to participate actively in the development process by creating user stories, providing continuous feedback, and making all necessary business decisions.
2. Programmers, often known as developers, are the members of the team who create the product. They are responsible to carry out user tests and implementing user stories. Since XP is usually associated with cross-functional teams, members' skills may vary.
3. Trackers or managers link customers and developers. It's not a required role, and any of the developers could do it. These individuals plan the meetings, regulate discussions, and maintain track of key performance indicators (KPIs).
4. Coaches can function as mentors for the teams to help them understand the XP practices. It's usually an external assistant or consultant who isn't involved in the development process but has expertise with XP and may help avoid errors.

# Values of extreme programming

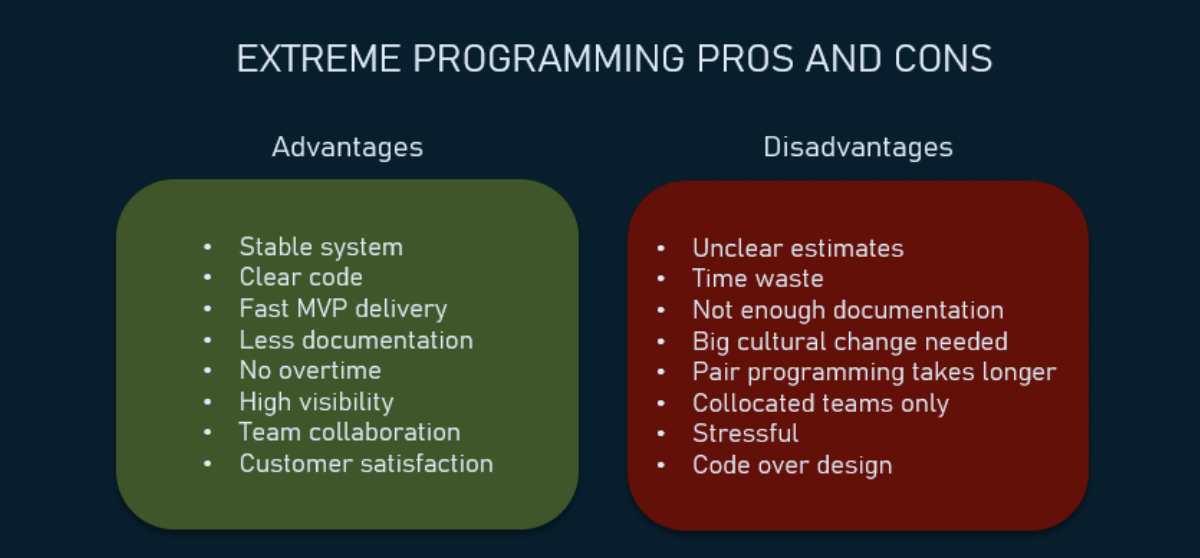
XP has simple rules that are based on 5 values to guide teamwork:

1. **Communication.** Everyone on a team works jointly at every stage of the project.
2. **Simplicity.** Developers strive to write simple code bringing more value to a product, as it saves time and effort.
3. **Feedback.** Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
4. **Respect.** Every person assigned to a project contributes to a common goal.
5. **Courage.** Programmers objectively evaluate their results without making excuses and are always ready to respond to changes.

# Principles of extreme programming

Most researchers denote 5 XP principles as:

1. **Rapid feedback.** Team members understand the given feedback and react to it right away.
2. **Assumed simplicity.** Developers need to focus on the job that is important at the moment and follow YAGNI (You Ain’t Gonna Need It) and DRY (Don’t Repeat Yourself) principles.
3. **Incremental changes.** Small changes made to a product step by step work better than big ones made at once.
4. **Embracing change.** If a client thinks a product needs to be changed, programmers should support this decision and plan how to implement new requirements.
5. **Quality work.** A team that works well, makes a valuable product, the team needs to come and regularly feel developing proud save resolve team of it.



### 4.3: Test-Driven Development (TDD)

TDD is based on the Agile manifesto concepts and Extreme programming. The test process, as its name implies, is what drives software development. Moreover, it is a structuring method that allows developers and testers to obtain optimized code that is resilient in long term.

## Phases of TDD

* **Create precise tests** To validate the functionality of specific features, developers must create precise unit tests. They must confirm that the test builds before running it. In the vast majority of circumstances, the test will fail. Developers are writing compact tests based on their expectations about how the product will operate, therefore this is a significant failure.
* **Correcting the Code** Once a test fails, developers need to make the minimal changes required to correct the code so that it can run successfully when re-executed.
* **Refactor the Code** Once the test has passed, look for any redundancy or code optimizations that could improve overall performance. Ascertain that refactoring has no impact on the program's external behavior.

## How TDD fits into Agile development

Agile development demands regular feedback to develop the expected product. Agile development can also be referred to as Feedback Driven Development.

During the development sprint cycle, there's a good chance that project requirements might change. To deal with this and produce solutions that are aligned with the changing needs of clients, teams require frequent input to avoid delivering unusable software. TDD is built to provide this type of early feedback.

## Benefits of Test-Driven Development (TDD)

1. Encourages the development of optimized code.
2. Helps developers in better analyzing and understanding client requirements, as well as requesting clarify when they are not clearly defined.
3. In the later stages of development, adding and testing new capabilities becomes much easier.
4. Test coverage is much higher in TDD than that in traditional development models.

## 5 - Agile Requirements

### 5.1: Requirements Management

The primary goal of Agile, a relatively new set of principles, concepts, and frameworks, is to help development teams and companies better serve their customers' needs. Above all, any Agile development project aims to deliver working software as fast as possible, with "working" meaning a product that can successfully handle the customer's problems. To be able to deliver a solution, one must first understand what the problem is. That’s why requirements management is at the heart of any Agile project.

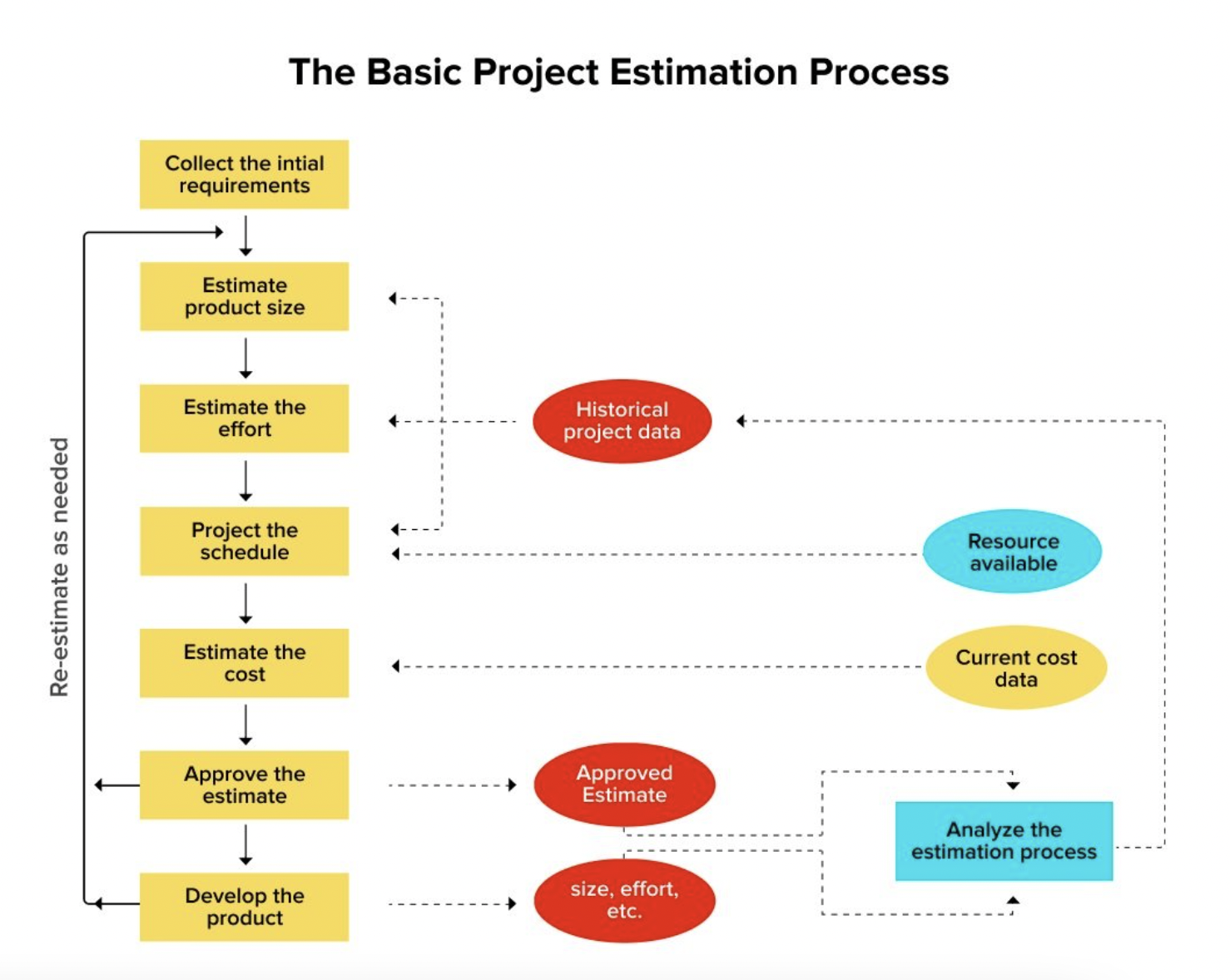
### 5.2: Effort Estimation

Agile development teams are prone to both overestimating and underestimating, resulting in varied development and launch times. Considering Agile estimation in the early stages of the process can help with accurate user story estimations and keep the team on track with timely deliverables, despite the complexity of the process.

The following are some of the key benefits of Agile Estimation techniques:

1. Improved Decision-Making  
     
    With story accurate, agile estimates the development team will be able to conduct effective backlog grooming sessions, which will benefit in precise sprint planning. Their user story delivery time will improve if they make informed decisions and plan well.
2. Better Coordination  
     
    Let's assume the effort for user story A is anticipated to be two weeks. User story B, on the other hand, requires a four-week estimation effort. Assume that both user stories are interdependent on one another. In that situation, the team must prioritize work so that both user stories are finished at the same time, resulting in better team coordination.
3. Better Risk Management  
     
    Budgets and timelines are often exceeded in software projects. Agile project estimate is an excellent way to minimize this risk. Agile product estimation helps in the estimate story points as well as the compliance to budgets, estimations, and project scope. The more precise the estimates, the more likely it is that the product will be delivered on time and in good condition.

### Stages of Agile Estimation

****

This is an exercise that typically takes two to four weeks, depending upon the project’s complexity.

The process includes:

1. Conduct Stakeholder Interviews
2. Define High-Level Product Backlog
3. Understand the Client and its Potential Customers
4. Prioritize Requirements
5. Prepare the Minimum Viable Product (MVP) Backlog
6. Estimate the Project Cost and Timeline

### Story Points

Traditional software development teams provide estimates in days, weeks, and months. Many agile teams, on the other hand, have transitioned to story points. Story points are a units of measure for estimating the total effort required to fully implement a product backlog item or any other piece of work. Teams assign story points based on the complexity of the task, the amount of work, and the risk or uncertainty involved. Values are assigned to help break down tasks into smaller chunks and address uncertainty more effectively. This helps teams understand how much they can do in a given amount of time, as well as build consensus and commitment to the solution over time.

A few reasons to use story points are:

* Non-project work that inevitably slips into our days: emails, meetings, and interviews that a team member may be involved in are not taken into consideration by dates.
* Dates have a strong emotional component. Emotional attachment is eliminated by relative estimation.
* Each team will estimate work on a slightly different scale, resulting in naturally varying velocity (measured in points). As a result, using velocity as a political weapon is no longer possible.
* You can quickly assign points without much dispute once you've agreed on the relative effort of each story point value.
* Story points are awarded to team members for solving tasks that are challenging rather than time consumption. This helps the team focus on providing value rather than spending time.

## 6 - Release Planning in Agile

### 6.1: Characteristics of Agile Planning

1. Agile development releases and fixed-length iterations  
     
    Releases and iterations are the two main delivery units in agile software development methods. A release is made up of multiple iterations, each of which is its micro-project. Features, defects, enhancement requests, and other work items are organized, prioritized, and assigned to a release. These work items are then prioritized and assigned to iterations within a release. The result of each iteration is working, tested, accepted software, and associated work items.
2. Agile development delivers working, tested software  
     
    The fundamental metric of progress for an agile development team is the delivery of working, tested features. Working features enable and improve team collaboration, customer input, and overall project visibility. They prove that the system and the project are on track.
3. Value-driven development  
     
    Agile development methods place a strong emphasis on providing business value quickly and consistently, as measured by running, tested software. This requires the team to focus on product features as the core planning, tracking, and delivery unit. The team keeps track of how many running, tested features they deliver from week to week and iteration to iteration.
4. Continuous planning  
     
    Planning must continue throughout the project, according to agile methodologies. Nobody on an agile project will take it for granted that the plan is workable until it can demonstrate its accuracy regularly.
5. Multi-level planning in agile development  
     
    Continuous planning is much more accurate if it occurs on at least two levels:  
   * At the release level, we define and prioritize the features that we must have, would like to have, and can live without by the deadline.
   * At the iteration level, we pick and plan for the next batch of features to implement, in priority order. If features are too large to be estimated or delivered within a single iteration, we break them even further.
6. Relative estimation  
     
    To speed up planning and eliminate unnecessary complexity, many agile development teams utilize relative estimation for features. Instead of estimating features across a spectrum of unit lengths, they select a few (three to five) relative estimation categories, or buckets, and estimate all features in terms of these categories.
7. Emergent feature discovery  
     
    Agile development projects prioritize and estimate items quickly, then refine details as necessary, instead of spending weeks or months detailing requirements before beginning development. Customers, testers, and developers collaborate to describe the features of an iteration in better detail. Additional features can be identified, but no feature is described in detail until it is prioritized for an iteration.
8. Continuous testing  
     
    We can deterministically measure progress and prevent defects using continuous testing. We crank out features that are up and running and have been thoroughly tested. This significantly lowers the chances of project failure in further stages.
9. Continuous improvement  
     
    Both the system and the project are being refined on a regular basis. We can alter our estimations and plans by reflecting on what we've done using concrete metrics like running, tested features, and more subjective measures. However, we also use the same method to continuously refine and improve the process itself.
10. Small, cross-functional teams  
      
     Smaller agile development teams have been proven to be far more productive than larger teams, with the ideal number of people ranging between five and ten. If you need to scale a project to include more people, try and keep individual teams as small as possible and coordinate efforts across teams. Scrum-based organizations of up to 800 employees are using a "scrum of scrums" method to project planning and coordination with great success.

### 6.2: Agile Release Planning

Agile release planning is a product management technique that involves planning incremental product releases. It is not the same as traditional software planning, which focuses on major releases.

When planning an Agile release, you prepare for staged releases and then break them down into several sprints or iterations.

Depending on the size of the project and the structure of your team, you may have several sprints running at the same time.

A sprint may end with a new product increment, although this does not always imply a product release.

Imagine writing a book: each sprint should end in a new draft. However, just because you have a finished draft for your client to review does not imply you need to make it available to the general public.

Release planning helps in planning which product increments and when they will be released to the market.

This approach helps your team in adapting to the unpredictability of software development.

It's also an essential part of the Agile Software Development Life Cycle.

### Purpose of Agile release planning

The purpose of release planning is to ensure that the product is always moving in the right direction and that logical releases are happening regularly. A release plan outlines upcoming releases soon but does not attempt to plan for the long term.

It's more detailed than a product roadmap. However, an Agile release plan does not outline the work that will be completed in each release. Instead, it groups iterations or sprints into releases.

Traditional executives are concerned that adopting Agile will result in each release being a random collection of features. A release strategy ensures that you always create a coherent version of your product.

It's an excellent tool for combining changes that will significantly improve the user experience.

### Steps to implement Agile release planning

1. Evaluate your product's vision and roadmap, focusing on specific outcomes.
2. Based on these results, expand and prioritize your product backlog.
3. Set a clear release target that incorporates all of the logical product backlog items.
4. Break your releases into several sprints or iterations.

## 7 - Iteration Planning

Iteration Planning is an event when everyone on the team decides how much of the Team Backlog they can commit to delivering in the next Iteration. The work is summarised as a series of committed Iteration Goals by the team.

### 7.1 Sprint as an Iteration

The terms sprint and iteration are similar. The standard duration for each is two weeks. On rare occasions, however, the work context may dictate that a one-, two-, three-, or four-week iteration is the preferred choice.

The iterations and their lengths should ideally remain consistent so that teams are aware of when specific ceremonies and milestones will take place. A sprint (or iteration) is just long enough to finish (develop and test) stories while still being short enough to pivot fast. Agility requires the capacity to pivot swiftly. The sprint or iteration time frame concludes with a system demo of new features in the staging environment, which is then ready for release.

### 7.2 Velocity and Capacity based Planning

**Velocity Based Planning**

The "Velocity" of an Agile team is the most important metric for planning. In each sprint, it is the quantity of work completed by the team. It aids the team in determining how far they may expect to make in a given sprint. Velocity is computed by adding all of the story points assigned to each finished user story by the sprint's end. It counts the number of items produced, but not the number of items produced.

The steps involved in Velocity-based Sprint Planning are as follows:

* Calculate the team’s average velocity (from last 3 Sprints)
* Select the items from the product backlog equal to the average velocity
* Verify whether the tasks associated with the selected user stories are appropriate for the particular sprint
* Estimate the tasks and check if the total work is consistent with past sprints

**Capacity Based Planning**

With traditional funding and planning methods, organizations often spend a considerable amount of time and resources tracking against plans based on historical information and business casing. This ultimately diverts focus away from achieving business outcomes. Adopting capacity-based funding and planning allows organizations to redirect their attention to delivering value and driving results.

Capacity-based funding distributes available funding across persistent, self-organizing teams. Funding is allocated based on the team's delivery capacity and the resources required to deliver the solution or product, rather than projects.

Capacity-based planning is performed at the team level, where teams prioritize work within the sprint and estimate hours available to commit to a specified set of activities. This method focuses on understanding the scope of products and future conditions to develop more accurate release schedules. It is recommended that teams continuously re-estimate to ensure delivery predictability.

### 7.3 Release Sprint Planning

The goal of Release Planning is to when various sets of usable functionality or products will be delivered to the customer which to enable the Scrum Team to have an overview of the releases and delivery schedule for the product being developed. With a longer-term plan, the team can align with the expectations of the Product Owner and relevant the project sponsor and provides us some answers for questions such as:

* When will we be done
* Which features can I get by the end of the year
* How much will this cost
* Discover critical dates and milestones
* Coordinate development plans of the dependent systems.
* Help us to balance business value and overall quality against the constraints of scope, schedule, and budget

**Patterns of Release Planning**

Many organizations have its own cadence regarding release of products to its customers. Some choose to release every sprint. Others group the results of multiple sprints into one release as shown in the Figure below. Still others release as soon as each feature is completed, a practice often referred to as continuous deployment or continuous delivery.

**Definition of Ready**

A release plan is a roadmap that reflects expectations about which features will be implemented and when they are completed. Depending on the development strategy, it may either be driven by functionality, in which the objective is to deliver a release once a predetermined set of functionality has been developed; or the planning may be driven by date, in which the release happens on a predefined check point. If the project is feature-driven, the sum of all features within in a release can be divided by the expected velocity. This will then result in the number of sprints needed to complete the requested functionality.

Why We Need A Release Planning?

Although, the actual dates of engagement in the agile world may have less precision as far as committed targets. However, a general delivery roadmap of releases could establish trust and expectation between your team and other stakeholders. Furthermore, Releases should take into account all the additional work that must be accomplished, such as updating the public website and training the customer support team.

Here is the main reasons for having a Release planning in Scrum your project:

* A communication device
* Planning tool
* Validates value versus cost
* Sets the overall context

## 8 - Executing a Sprint

Sprint Execution is performed during each Sprint by the Scrum team to meet the Sprint goal. This tutorial centers around the standards and practices that guide how the Scrum team oversees, plans, performs, and discusses during Sprint execution.

Every Sprint, the team implements a project in small chunks and performs all the necessary tasks to deliver a potentially shippable product at the end. The team’s work is based on the Sprint goal and Sprint Backlog.

In Sprint Execution, the majority of the time is spent during each Sprint. Sprint Execution starts after Sprint Planning and ends before Sprint Review. For a two-week Sprint, the duration of Sprint Execution lasts for 8 to 10 days.

During Sprint Execution, the Scrum Master, development team, and Product Owner should be present. The development team members decide the most ideal way to meet the Sprint goal. The Scrum Master mentors, facilitates, and expels obstacles that are blocking the progress of the team. The Product Owner has to be there to solve queries on product requirements, review the work in progress, and give inputs to the team. The Product Owner can also be called sometimes to discuss the adjustments in the Sprint goal and to confirm the acceptance criteria.

Sprint goal and Sprint Backlog are the inputs to the Sprint Execution process. These inputs are generated during a Sprint Planning. Sprint Execution process includes task planning, performing, managing the tasks, attending daily stand-ups, and communicating with the Scrum teams. The outcome of Sprint Execution is a potentially shippable product increment, formed from a list of product backlog items by meeting the team members’ definition of done.

### 8.1 Sprint Ceremonies

The short answer to what scrum ceremonies are, is meetings. But these are not just regular team meetings where you come together and discuss some stuff, hoping that something useful eventually comes out of it.

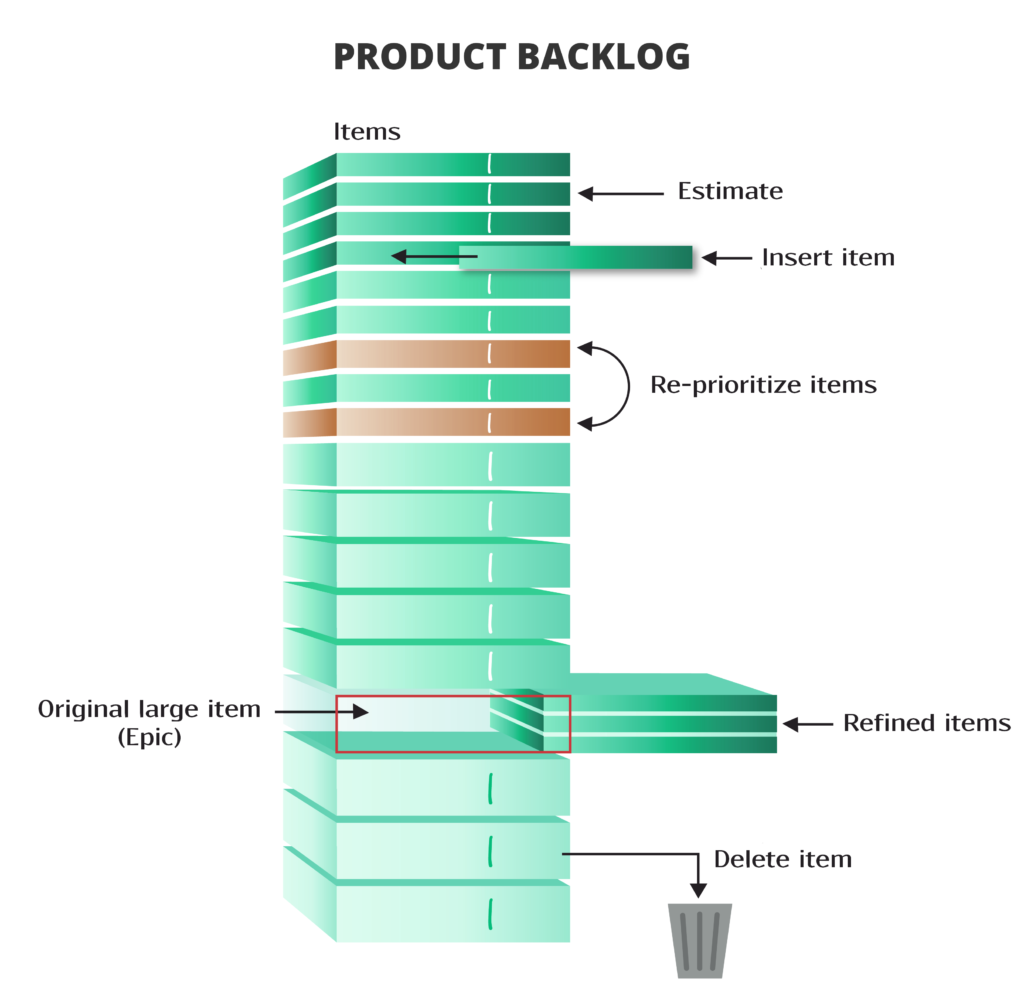
These are the five key scrum ceremonies:

* Backlog grooming (product backlog refinement)
* Sprint planning
* Daily scrum
* Sprint review
* Sprint retrospective

Apart from Backlog Grooming, the rest four ceremonies are meetings. As already mentioned, each one of these meetings has a special goal and timeboxing, so let’s cover them one by one.

**Product Backlog Grooming**

Backlog grooming is the only ceremony in scrum that doesn’t have a defined time box or even a frequency. It is, however, a critical responsibility of the product owner with the help of their team to add new items to the list and order them based on their priority. At the same time, outdated, redundant or non-valuable items should be removed from the backlog to keep it clean, valuable and actionable.



Product Backlog

The product backlog is a list of everything that is to be developed for your product or project. It can include new features, bug fixes, optimizations, or pretty much anything that somehow benefits the customer. If you have multiple scrum teams working on the same product, every team has their own sprint backlog, yet all the teams usually share the same product backlog.

The product backlog needs to be constantly updated and refined based on stakeholders’ feedback, new ideas, sprint review meeting results, emerging market opportunities and so on. The more often you update and refine your product backlog, the more flexible, dynamic, and ultimately competitive your company will be.

**Sprint Planning**

All the work in scrum is cyclic: it happens during recurring, time-boxed periods when scrum team members are focused on delivering something of value to the customer. These iterative periods are known as sprints, and they have the same duration each time, somewhere between one week and a month.

Sprints follow on one after another, without any pause between them. Every team sets the duration of their sprints themselves. But to start a sprint, the team must know what they are going to develop. This is exactly what the sprint planning ceremony is for.

The team gathers and starts to pull items from their product backlog into the sprint backlog. The team needs to carefully consider how many items to take on, as they will make a commitment to completing each item on the sprint backlog before the sprint ends.

**Sprint Backlog**

The sprint backlog is a list of items (also known as user stories) that the team is going to deliver during the sprint. During the sprint planning, the team selects the items in order of priority from the product backlog to form the sprint backlog.

Teams usually break down sprint items into smaller chunks, known as tasks. That way they can share the work out among the team, and track their progress in a more granular way to ensure they stay on track to complete the item before the end of the sprint.

Only the development team (NB excluding the product owner) can decide how many items they can commit to during the sprint, and only the scrum team (NB including the product owner) can make changes to the sprint backlog during the sprint.

During the Sprint planning, the team takes into account their past performance, which allows them to more accurately forecast how many items they can finish. Team members should also account for other factors such as holidays and time-off, as well as other commitments that may reduce the amount of time they have. If it is the first sprint that the team has worked together (i.e. they have no past performance), they tend to go on gut feeling, or plan to only do small, simple tasks which they can more easily estimate in time.

Only the development team can decide which items to take into the sprint, and while the product owner attends, nobody outside of the scrum team can participate in sprint planning meetings.

“The more often you update and refine your product backlog, the more flexible, dynamic, and ultimately competitive your company will be.”

Finally, during sprint planning, the team also collaborates to choose a sprint goal, which helps team members to clarify the purpose behind this particular sprint. Choosing a sprint goal is key for a team’s productivity, so scrum teams should not neglect it.

Sprint planning summary

* Goal: to come up with the items and goal for the new sprint
* Participants: the entire scrum team (developers, product owner, scrum master)
* Frequency: once per sprint, at the start of the sprint
* Duration: 2 hours for every 1 week of a team’s sprint duration

For effective sprint planning: experienced teams break down sprint backlog items into smaller chunks (tasks) to better evaluate how much time they need to finish them.

**Daily Scrum**

Daily scrums, or daily standup meetings, are a driving force for every sprint.

The development team and scrum master gather each day for a very brief period of time (limited to 15 minutes) to discuss their current progress and any blockers that prevent them from finishing their tasks. The product owner can also attend to hear how things are going, and to offer guidance or clarification if the team needs it. But in essence the meeting is for the solution development team, and not an opportunity for the product owner to put more work on them, make decisions about how the team should work, or comment on the pace of progress.

Below are the three typical questions every development team member answers during a standup meeting:

* What have you completed since the last meeting?
* What do you plan to complete by the next meeting?
* What is getting in your way?

The goal of daily scrum meetings is to look at how things are going and to make a plan for the next 24 hours that will keep the team on track to complete the work they committed to by the end of the sprint. Traditionally, standups are in-person meetings with team members physically standing in the same room. Standing forces team members to keep what they say to the point, which means the team can keep the meeting to the 15-minute timebox.

Unfortunately, standing in the same room is impossible for distributed teams that practice scrum. This often leads to inefficient virtual meetings and in turn to a general distrust in Agile methodology. Fortunately, remote teams can work around that with asynchronous daily standups.

Daily standup summary:

* Goal: to share progress, check if they are still on track to successfully complete the sprint, and make a plan for the next day of work
* Participants: Scrum team members (including the scrum master). Product owner is optional.
* Frequency: daily
* Duration: 15 minutes max

For effective daily standups: scrum master should be willing to tell team members when they are getting off-topic, talk about the items in order of priority (rather than just whoever wants to go first), use asynchronous standup meetings if your team struggles to keep daily standups focused and efficient.

### 8.2 Sprint Reviews and Retrospectives

**Sprint Review**

The sprint review is a meeting that takes place at the end of the sprint.

The sprint review ceremony has three main objectives:

* Team demonstrates their finished work to the product owner and stakeholders
* Team discusses any feedback and business context for this project
* Team updates the product backlog based on the discussion

Many scrum teams tend to neglect items #2 and #3 and focus only on the demo part of the meeting, thus incorrectly referring to the sprint review as the “sprint demo”.

Yet discussing business context is crucial because it helps teams to decide what are the most valuable things to work on next. Those discussions may include market developments, user feedback, and company goals.

Another important requirement is that, at the end of the sprint, work is finished and ready for the customer to use.

Product Increment

A product increment is a new or updated, usable version of the product.

In other words, it’s the tangible output of your sprint. A new feature, an updated interface, a new integration, or even a fixed bug can form part of a product increment as long as you can clearly demonstrate its value and apply the increment to your project.

Sprint review summary:

* Goal: to present and discuss finished work, provide business context for future developments, and update product backlog
* Participants: everyone (the scrum team, stakeholders, any interested party)
* Frequency: once per sprint, at the end of the sprint
* Duration: 1 hour for every 1 week of a team’s sprint duration

For effective sprint reviews: make sure your team members take time to prepare what they will show so stakeholders can easily understand their finished work and engage fully in the discussion.

**Sprint Retrospective**

Sprint retrospectives are critical for scrum teams that want to constantly improve the efficiency and quality of their work. All members of the scrum team attend sprint retrospective, which takes place after the sprint review. The team:

1. Inspect how the sprint went in terms of people, relationships, process, and tools.
2. Identify what went well and opportunities for improvement.
3. Create a plan with specific improvements to be carried out during the next sprint.

3 is crucial and often neglected by inexperienced teams. In order to bring value, retrospectives need to be actionable. Simply uncovering areas of improvement without doing anything won’t bring results.

Sprint retrospective summary:

* Goal: to uncover opportunities for improvements and create a specific actionable plan to carry them out
* Participants: all members of the scrum team
* Frequency: once per sprint, after the sprint review
* Duration: 45 minutes for every 1 week of a team’s sprint duration

For effective sprint retrospectives: focus on opportunities, not blaming other team members, vary the format so that it doesn’t become predictable and boring for the team members.

## 9 - Agile Metrics and Tools

### 9.1 Overview of Agile Metrics

Metrics are nothing but standards of measurement. Agile metrics are standards that help a software team in monitoring how productive a team is across the different phases of the SDLC.

Agile metrics are an essential component of the development process. For companies or teams that work on the agile framework, agile metrics help in assessing software quality.

By measuring how productive a team is, agile metrics help keep the team performance in check. If there are any loopholes, they expose them at the initial stages. Since the data and its usage are measurable, it’s easier to work on the shortcomings with the help of these metrics. For example, velocity metrics can help you track your team’s output.

The classification of Agile metrics isn’t set in stone — it’s always changing and restructuring. Still, over the years, we’ve seen the rise of three main kinds of Agile metrics in various Agile frameworks.

* Kanban metrics — these metrics are based on measuring invested time (cycle times) and delivered results (throughput), and their ratio.
* Scrum metrics — metrics, which are focused on planning and understanding the workflow and demonstrating how much work was performed in a given period;
* Lean metrics — the continuous measurement of production efficiency and product quality with technical assessment by testing features, checking for possible errors, and foreseeing negative effects.

### 9.2 Tools for Agile Project Management

As a project manager, you need agile tools for metrics to ensure that the product is not only finished, but also of high quality. These use various KPIsto show you where the development has problems and you need to readjust accordingly. The result: you produce a result that is a bull's eye. That is better than that of the competition.

1. JIRA Atlassian is a tool for agile metrics that is mainly used by agile software development teams. You can use Jira to create Kanban and Scrum boards, track bugs, view important features and monitor them. In the process, it creates a roadmap of the entire project. This gives you a nice overview of how all tasks are connected. To evaluate agile KPIs on an ongoing basis, Jira automatically reports on the following metrics, among others

* Velocity,
* Sprint Burndown,
* Epic and Release Burndown,
* Control Chart,
* Cumulative Flow Diagram and
* Defects.

This way, you can use Jira to organize your projects, formalize workflows, and optimize KPIs.

1. Trello is an online tool that allows you to monitor your project progress. For this purpose, Trello visualizes projects in a single board. This way, you always have an overview of which tasks have progressed. The basic version does not include an agile tool for metrics. However, with paid extensions, you can add agile KPIs such as

* Burndown Charts,
* Cycle Time,
* Cumulative Flow Diagram,
* WIP Age and
* Burnup Charts

into Trello.

1. Pivotal Tracker is a project planning tool for software development teams. It is built to keep your agile team on track to hit their sprint goals. It’s sprint planning guides you with breaking down and prioritizing projects into manageable, actionable chunks for your team to keep the momentum going. Agile features in Pivotal Tracker:

* Stories
* Estimation
* Backlog
* Iterations
* Velocity chart

## 10 - Quality Management in Agile

### 10.1 Managing Quality in Agile Project

In an agile project the whole team is responsible for the delivery of a quality product throughout the time that they are working on the project. The project team have responsibility for deciding how the product will be created, how activities will be divided amongst its members and how their work will be organised. They have control over the product process rather than being directed in what they do by a project manager.

The work of the project team is designed so that the needs of the client or customer are constantly being reviewed and changes that may be required can be made easily. The product owner is responsible for representing the needs of the product and ensuring the members of the project team are aware of them and that the work done is focused on them.

### 10.2 Managing Risks in Agile

Project failure remains a serious issue in project management. Failure rates are very high and the rise in popularity of the agile methodology is due to a belief that it will make project success more likely, partly due to their approach to managing risk.

Predictive projects manage risk by identifying, assessing and planning mitigation of it at the beginning of the project. Continuous monitoring of risk and updating of plans should take place as the project proceeds. However, this is often done badly, if at all, partly because change of plans in a predictive project is often cumbersome.

In agile projects, the focus on regular completion of deliverables that provide value to the client or user as well as the delayed planning of detailed activity through iteration, make it easier for risk to be dealt with. It is important also to note, though, that agile projects need to consciously ensure that risk is being managed effectively in the project. It needs to consciously be a part of the project team’s regular work.

How Agile Is Intended to Minimise Risk

Agile Project Management has replaced the Waterfall approach in many organisations, especially in the technology sector. This is because it has been found to have substantially higher rates of success.

The agile approach is intended to enable changes to be incorporated in projects easily. Predictive projects are more difficult to change and are more vulnerable to large scale failure. Project activity normally leads to a final product at the end of the project that requires successful completion of a complex series of tasks. As the project proceeds and more work is done and resources consumed, the cost of the project increases and the benefits, if the project is successful, are only realised when the project is completed. Risk increases through the lifespan of the project and is greatest at the end (when everything is lost if an insurmountable problem arises).

Common Risks in Agile Projects

While agile projects are designed to minimise risks that does not mean they don’t exist. Deloitte have identified the following as common problems that agile projects encounter:

* Using agile when the project is not suitable
* Not sticking to agile principles
* Reducing governance and oversight
* A break in the flow of requirements
* Bypassing change management processes
* Poor or non-existent forecasting

## 11 - Agile Myths and Pitfalls

### 11.1 Common Mistakes and Myths in Agile

**Common Mistakes in Agile**

1. Trying to implement too quickly

Enthusiasm for change toward more Agile project management practices can be motivating, but it can also result in teams trying to run before they can walk. Before hoping to be successful, teams need to be clear on what exactly they are changing and how these changes will be put into practice. A rush to get things started will only cause confusion and most likely failure to actually adopt Agile, with everyone just going back to what they did before and branding Agile project management a disaster that doesn’t work for them.

2. Not getting leadership buy-in

One of the biggest challenges of Agile project management is getting those in leadership positions on board with the fact that their team members will now be empowered to make decisions without them. While some will be delighted to see their team take on greater responsibility, others might see this as undermining their authority or even the necessity of their role altogether.

3. Putting too many people on your team

Departments used to running big teams can have trouble cutting things down to size for Agile on a project-by-project basis. This makes decision-making and collaboration cumbersome and negates the benefits of Agile’s greater decision velocity.

4. Trying to do too much in your sprints

Another common issue when teams first adopt Agile is trying to replicate the “target beating” philosophy of more traditional workplaces in their sprints. While it’s great to have targets in general, a sprint’s primary goal is to get everything done, then assess whether everything was achieved and if not, why? Trying to crowbar too much work into a sprint to “motivate” the team to push itself more will only lead to sprint deliverables being missed and team morale being thrashed by constant (perceived) failure.

5. Draining productivity through shoulder tapping

Though it’s never a formal work practice, a lot of work in most offices is assigned through “shoulder tapping” – that is, when a manager (or even another team member) approaches someone and asks them if they could work on something. While this is a natural enough way to delegate tasks, it can also undermine one of the core tenets of Agile: decentralized decision-making through empowered teams. In other words, truly Agile teams shouldn’t need to rely on a central shoulder-tapper, but rather the collective initiative of motivated, accountable individuals with an equal stake in the project’s success.

**Common Myths in Agile**

1: Agile Is A Set Of Project Management Frameworks

Reality: Agile is primarily a culture, a way of thinking and acting.

The biggest and most common mistake and the reason that many fail at implementing Agile in organizations comes from focusing on the DOING Agile without working on the BEING Agile. Implementing Scrum, Design Thinking, Hackathons, Lean, Kanban and other Agile frameworks will not be sufficient to be Agile.

2: Leaders Are Not Needed In Agile

Reality: True…and false.

Agile needs leaders, but not where they might usually spend their time and energy.

Their role is to drive and foster the appropriate ecosystem and culture. They must genuinely inspire themselves first and then their people with a compelling Purpose, Vision, and Strategy that can guide decisions and actions. This is not a minor role. An Agile organization could not exist without these leaders. In Agile, coaching leaders to empower their people by decentralizing decisions, control, and accountability to the point closest to action.

To put it in other terms, leaders move from the pilot seat to the co-pilot one.

3: Agile Is Better And Faster. Its Role Is To Increase The Speed Of Decisions And Actions.

Reality: Speed of decision and action is part of a predefined daily and weekly planning but is not a goal. Value delivery to customers comes before timing.

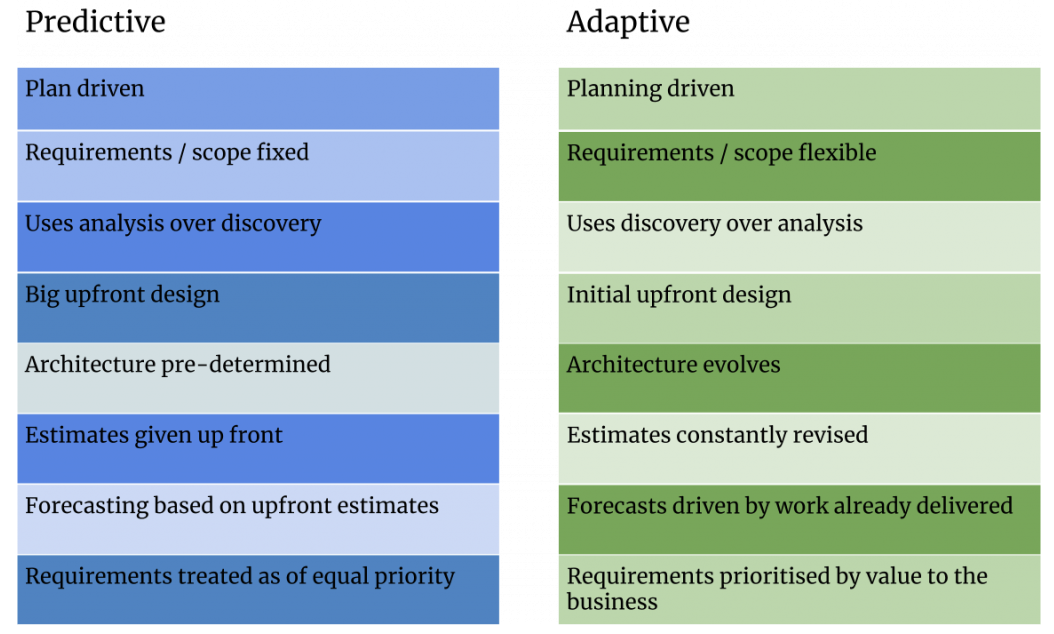
4: Agile Is About Producing More, Quicker, And Cheaper

Reality: Big mistake. Agile optimizes value delivery and customer satisfaction first, not just productivity and efficiency.

5: Agile Is Permanent Instability Management

Reality: Agile’s pre-defined cadence and framework are highly predictable. You know in real-time how the team is tracking against objectives with daily baby steps, one at a time, with clear objectives. This approach makes manageable permanent changes and instability with iterative adaptations, learning and improving from mistakes/successes with clear metrics from customer feedback.

### 11.2 Predictive Planning vs Adaptive Planning



### 11.3 Distributed Agile

The main point in adapting Agile for a distributed team is not sticking to each point of the Agile Manifesto or trying to implement each of the basic principles of Agile. By its nature, distributed cooperation will have its effect on the development flow, however, this does not mean that Agile will not work.

**Agile team size and team roles**

When applying the Agile methodology in a distributed team, it is important to set up the team so that to achieve the maximum effect. All references say that the optimal size of an Agile team is 3 to 9 members. A larger team may affect the communication quality, as even in a face-to-face daily meeting, a person has a certain limit of what they can remember at the same time. Learn more about Scaled Agile Framework (SAFe).

Moreover, in a distributed team, you will anyway face the problems of time zones and the members’ personal schedules. Thus, we recommend that you keep the distributed team size to the recommended number to facilitate the communication organization.

**Workload distribution**

Of course, the recommendation to distribute the workload evenly between the team members works not only for distributed teams but for co-located ones, too. However, in a distributed team uneven workload can have the most severe consequences.

When an overloaded team member delays the flow for other colleagues, it can become even longer due to time differences or schedule mismatch. Therefore, an overload can threaten the timely delivery for the entire team. At the same time, people with too little work may become poorly motivated and not willing to commit their effort to the project.

If your team structure includes both in-house and remote colleagues, make sure the remote members do not find themselves in the position of subordinates. They are not “helping” the head office, they are doing their share of work for the benefit of the common project.

**Issue and backlog management**

The daily sync-ups that are prescribed by the Agile methodology are intended, among other things, to detect the issues that block other team members’ work. In remote Agile teams, this problem is of even higher importance, as a blocker can not only halt the developer’s work but make them wait until the colleague who should resolve the issue starts their working day on the other side of the globe. As the result, a lot of precious time is wasted.

To avoid such bottlenecks, adopt the practice of discussing not only the today’s issues but also those that are planned for tomorrow. This way, it will be easier to see what may block the development and take proactive measures.

Smart management of the current issues and the project backlog can reduce the dependencies in the team to the minimum. This way, you can achieve higher efficiency and better productivity of your entire team.

**Pair programming**

At first glance, pair programming is outright impossible in a distributed team. Pair programming means two people working on the same station – one is writing the code while the other monitors it, makes suggestions, asks for clarifications, notices mistakes. Pair programming is one of the core methods of Agile intended to improve the code quality.

In fact, you can set up productive pair programming in a distributed team, too. Today, there are quite a lot of conferencing tools for any budget. Most of them have the screen-sharing feature which will be the effective replacement of the side-by-side programming. Slack, Skype, Zoom – all of them can be used to organize pair programming. You can read more on the tools used in managing distributed teams in our blog.

**Agile tools for distributed teams**

In setting up an Agile development distributed team, make sure you provide proper communication and collaboration tools so that your team members have no problem in reaching each other or attending daily meetings.

Since your company develops software, you are bound to have GitHub as your platform for code building and review. Use Google Drive to manage your project documentation or opt for Microsoft SharePoint if you need a more comprehensive collaboration package with enhanced functionality.

Ensure your virtual team communication by setting up an instant messenger, such as Slack or Skype. The same tool can also be used as a means of your daily meetings.

**Cultural differences**

Inviting professionals from different geographical regions gives tons of benefits for the company – from cost saving to leveraging the different approaches to software development. At the same time, managing a distributed team requires understanding the cultural specifics of each region and adapting them to the common flows.

## 12 - Ensuring Agile Success

Agile Scrum is just a framework which needs to be followed within its operating context. A team should not get trapped between the framework’s boundaries. Following are some of the Scrum which indicate that a team is trapped between the processes.

* Team religiously forced to start/ finish the ceremonies without having right results or seeing improvement. Having time-bound and scheduled ceremonies (like standups) are important but it should end on a positive note with evidential, useful and actionable/ improvised outcomes
* Team disinterested to attend ceremonies, especially Daily Scrum, or it (daily scrum) is seen merely as a status update to Scrum Master/ Product Owner. Chaotic environment and team’s unwillingness to attend retrospectives. Unsatisfied business users and frightened or defensive product owner during sprint reviews
* No engineering practices. No automation to code reviews, integration, build, unit test and software delivery. Required frequent and repetitive tasks are done manually most of the times
* Virtually no improvement or sustainable pace even after completing multiple sprints. Team’s vital metrics, such as velocity, widely fluctuate or have a declining trend. Because of this, the performance and productivity cannot be predicted
* Repetitive instances of failed (or killed) sprints, accumulated technical debt, inability of team to achieve sprint commitments and jeopardized long-term product goals
* Team’s unwillingness to participate in backlog grooming and reluctance to estimation & planning. Features (or user stories) are broken forcefully to fit into sprint length
* The team is not organized as feature teams. There are too many dependencies on cross teams or on specialized skilled person/ role such as User Experience, Architects, DBAs, Testers etc. Testers and developers not working in coordination
* External stakeholders feel ignored during legitimate and critical inputs and thus, start losing faith in the development team. They struggle to provide feedbacks and hence, adopt pressure techniques

Go Beyond Scrum for Being Agile!

* Scrum without Lean and Agile values will fail!
* It might appear to be okay if a team follows Scrum but, to be successful in the long–run, the team must embrace Lean and Agile values & principles together. Continuous focus on adhering to the guiding principles behind the practices being adopted can form the basis for delivering business values and customer delight. Following are some of the key aspects to be considered:
* Agile focuses on – Early and continuous delivery, incorporation of changing requirements, customer satisfaction, shorter delivery timescale, increase in customer feedback loops, open communication, customer collaboration, involvement of business & cross-functional teams, technical excellence, sustainable development pace, inspection, adoption to continuous improvement etc.
* Lean emphasizes on – Optimizing the whole system, reducing waste, mapping value streams, improving continuous flow, minimizing work in progress, applying pull & just-in-time mechanism etc.
* Embrace other methods & practices
* There are many other agile practices that can help to realize Scrum with its true intent. These whole bunch of agile practices addresses product development lifecycle from various aspects such as product portfolio management, requirement engineering, software process engineering etc.
* “Scrum method is definitely the foundation and enabler to agility but it’s not enough if it’s not being practiced along with other agility practices.”
* Here are some of the leading agile practices:
* XP & software engineering practices – Scrum does not prescribe any practices but it supports modern/ effective engineering practices such as TDD, ATDD, Refactoring, Pair Programming, Continuous Integration etc.
* Kanban – It is mainly a product portfolio management practice and helps in visualization of work, limiting work in progress, and effectively improving system flow.
* Numerous other agile practices can be considered in different aspects for the realization of benefits. Please refer to the post ‘Four Facets of Agile – Part 2 (Agile Practices)’ in the references below for the comprehensive list of Agile practices.
* Build Agile culture & mindset
* “There is no magic wand to build Agile culture and mindset.”
* Scrum by itself cannot succeed without having agile culture and mindset in the organization. All stakeholders like customer & product management and development teams should have undergone agile training and should be willing to embrace agile values and principles. With time, a consistent approach for improving process/ practice maturity collaboratively as well as knowledge sharing sessions can help an organization build the right culture and mindset. Please refer to the post ‘Four Facets of Agile – Part 1 (Agile Mindset)’ in references below for more details on agile culture and mindsets.
* Agile coach and coaching model
* An agile coach and a well-structured agile coaching model enable organizations/ projects to achieve a quick and stable turnaround in order to create values by using agile best practices. It helps not just in creating awareness but also in handling change management for agile transformation/ adoption and execution. By implementation of best practices, it improves continuous delivery maturity. Agile coach should be associated with each project team for coaching and supervision, especially when an organization is transforming from waterfall to agile methodologies. Additionally, when a project team is new to agile and Scrum Master or Product Owner are performing their roles for the first time, associating with an agile coach can highly benefit them in executing their duties efficiently.
* You need support by executives/ management
* Scrum essentially not only needs executive sponsorship and management support but also requires continuous involvement throughout the agile adoption and transformation journey. Unless executives/ management teams understand pain and challenges and extend their support for organizational changes, it would not be easy to succeed. Teams may struggle to perform and deliver efficiently. In order to succeed in the agile scrum, a commitment is required across the hierarchy in embracing lean-agile practices. An organization should ensure that its team has a requisite environment available (e.g. development, testing, pre-production/ staging) that is equipped with required tools and technologies. This is essential for effective collaboration, automated testing and for the entire delivery pipeline.
* Reinvent Agile as the team goes along
* Improve on processes, practices, communication & collaboration and inspection & adaption. It is recommended to have a coach to guide and direct with respect to agile methodology. He/She can also help to perform audits and reviews so as to help the team in adapting the best practices and filling any process gaps. This can definitely help a team to sail through the agile journey even in difficult times.
* Business’s commitment to adopting agile methodology is also required
* Customer’s business team should also understand and appreciate adaption to agile and should be committed to actively participate in agile practices. If required, they also need to be trained on agile methodologies so that they are aligned with the development team and can help to practice agile on their part in terms of proactive and timely reviews, acceptance, and feedback.
* Team should possess the required technical skills
* Development team must be competent enough to skillfully engage agile processes, practices, tools and technologies to achieve optimum benefits. Just being trained on agile/ Scrum and having knowledge of frameworks is not enough. A team should practice it in real-time project scenario to reap maximum advantages.

### 12.1 Managing Change

The 3 main actions that you need to take when managing a change in an Agile environment are:

* Educate the organization on Agile
* Engage sponsors more effectively
* Test new approaches

### 12.2 Evolution of Agile with Times

There has been a growing need for a new way of organizing work for quite a while, since the 90s. This idea was especially true with software development teams that needed to account for changing customer input, be able to adapt, and be less dependent on a predefined plan. As a result, in 2001, the Agile Manifesto has come about.

Stating four core values and 12 principles, the Agile Manifesto laid down the foundation for what we know as Agile today. And while some of its signers and applications do contradict one another. What the main ideas stand for is aligned.

A declaration is an essential point in the Agile movement. It gave the varied applications and attempts, an umbrella term and core ideas to stand behind.

After the authors got back from Snowbird, Ward Cunningham posted the Agile Manifesto, and later the 12 Principles online at AgileManifesto.org. People could go online and sign it to show their support.

Agile Alliance was officially formed in late 2001 as a place for people who are developing software and helping others develop to explore and share ideas and experiences.

While some of the Agile applications were already functioning before the Agile Manifesto, they truly solidified after. The software development teams finally had a more flexible alternative to traditional project management. It offered an environment that was organized around the product and not the process.

More and more of the software development teams started using varied Agile applications like Scrum, Kanban, Extreme Programming, and other Agile methodology frameworks. At the same time, the Manifesto and Agile Alliance gave guidance and simplified the transition for those wanting to join.

At this stage, Agile was mostly used and adopted by software development teams. They used Agile frameworks locally to have better control of their processes, while still complying with traditional project management practices on the organizational level. However, as the positive influence of Agile became more and more apparent, the interest of the outside teams grew.

Once Agile had a good following and showed results with the software development teams, others seemed to notice. Finance, Marketing, Design, and other teams saw value in this way of organizing work and started implementing Agile as well. While at first, it seemed Agile could only benefit software developers, this was quickly disproved. And this new notion kick-started the agile evolution and expansion to other sectors.

As Agile became more widely known, an ecosystem formed that included the people who were doing Agile software development and the people and organizations who helped them through consulting, training, frameworks, and tools. So called Agile coaches were born and retain their relevance also today.

Teams with various backgrounds and expertise adopted Agile frameworks successfully. What pushed this adoption forward, though, was the courage to innovate and change the structures themselves. The teams saw how software developers worked and adapted the frameworks to work for their needs, all the while keeping up with the Agile Manifesto.

## Questions Solved

### 1: Ending Sprints with non-shippable functionality is an *anti-pattern* to becoming more agile. Why is this counterintuitive in an agile setup? Suggest one method of remedying this

**Ans:**  By the end of each iteration, a Scrum team is expected to produce a Potentially Shippable Product. As we know Agile methodologies emphasize “Working Software over Comprehensive Documentation”. When we talk about working software, it is both complete and potentially shippable.

Agile methodologies emphasize working software that is both complete and potentially shippable due to the following 3 reasons:

* It encourages feedback
* It helps a team gauge its, progress
* It allows the product to ship early if desired

Hence, the scenario in question is counterintuitive in an agile setup.

One possible method of remedying this would be to adopt SCRUM. Scrum is a framework that can be used for many situations. For teams that release fast (every Sprint or multiple times per Sprint) and for teams that build increments that take more Sprints to release.

Benefits of SCRUM in this scenario will be:

* Quicker release of usable products to users and customers.
* Higher quality.
* Higher productivity.
* Lower costs.
* Greater ability to incorporate changes as they occur.
* Better employee morale.
* Better user satisfaction.
* Being able to complete complex projects that previously could not be done.

### 2: Explain three ways by which SCRUM supports continuous inspection and adaptation.

Ans:

Three ways by which SCRUM supports continuous inspection and adaptation are:

* **Transparency**: This means presenting the facts as is. All people involved—the customer, the CEO, individual contributors—are transparent in their day-to-day dealings with others. They all trust each other, and they dare to keep each other abreast of good news as well as bad news. Everyone strives and collectively collaborates for the common organizational objective, and no one has any hidden agenda.
* **Inspection**: Inspection in this context is not an inspection by an inspector or an auditor but an inspection by every- one on the Scrum Team. The inspection can be done for the product, processes, people aspects, practices, and continuous improvements. For example, the team openly and transparently shows the product at the end of each Sprint to the customer to regularly gather valuable feedback. If the customer changes the requirements during inspection, the team does not complain but rather adapts by using this as an opportunity to collaborate with the customer to clarify the requirements and test out the new hypothesis.
* **Adaptation**: Adaptation in this context is about continuous improvement, the ability to adapt based on the results of the inspection. Everyone in the organization must ask this question regularly: Are we better off than yesterday? For profit-based organizations, the value is represented in terms of profit. The adaptation should eventually relay back on one of the reasons for adopting Agile—for example, faster time to market, increased return on investment through value-based delivery, the reduced total cost of ownership through enhanced software quality, and improved customer and employee satisfaction.

### 3: A customer comes to you and discusses his initial requirements. The requirements are evolving, and the customer is interested in being part of the development process. Nevertheless, the customer expects to see the software artifact only at the end of the project timeline, which is here ten months. As an agile consultant, suggest the best way to handle this situation and name the methods you would use.

**Ans:** Involve customers in various Agile ceremonies like Product Backlog grooming, Sprint Planning, Daily Standups, and demos. This way the customer can provide continuous feedback and the development progress is transparent to them. This way any changes that the customer suggests can be incorporated into the backlog.

### 4: What are the roles of the product owner, the scrum master, the development team, and the agile mentor? Instantiate the roles explained previously with the help of an e-commerce development project example.

Ans:

Assuming an e-commerce development project, as a website to buy shoes.

A brief description of the roles are given below:

1. **Product Owner:**

The product owner has experience running e-com projects, understands the e-com market trends, represents the stakeholders of the project, and is responsible for setting the direction for product development of the website. The Product Owner understands the requirements of the project from a stakeholder perspective and has the necessary soft skills to communicate the requirements to the product development team. The Product Owner also understands the long-term business vision about the web commerce market and aligns the project with the needs and expectations of all stakeholders. The key responsibilities of a Product Owner include:

* Scrum backlog management
* Release management
* Stakeholder management

1. **SCRUM Master:**Scrum Master ensures team coordination and supports the progress of the project between individual team members. The Scrum Master takes the instructions from the Product Owner and ensures that the tasks are performed accordingly.

The role may involve:

* Facilitating the daily Scrum and Sprint initiatives
* Communicating between team members about evolving requirements and planning
* Coaching team members on delivering results
* Handling administrative tasks such as conducting meetings, facilitating collaboration, and eliminating hurdles affecting project progress
* Shielding team members from external interferences and distractions

1. **Development Team:**

The team members within the Development Team are individuals with responsibilities including but not limited to product development. The team takes cross-functional responsibilities necessary to transform an idea or a requirement into a tangible product for the end-users. The required skills might be wrapped up in one or more dev team members based on required skills for building an e-commerce application (website to buy shoes online):

* + ERP Subject Matter Expert
  + Cloud Engineer
  + Developer
  + Tester
  + UX specialist

Not every member may be an engineer but maybe a part of the team if their skills are required for the project to proceed at the required pace.

1. **Agile Coach:**

Agile coaches help train corporate teams on the agile methodology and oversee the development of agile teams to ensure effective outcomes for the organization. They are responsible for guiding teams through the implementation process and are tasked with encouraging workers and leadership to embrace the agile method

### 5: Explain *progressive elaboration* of requirements in different stages of an agile project. List the items shaped by such progressive elaboration in all those stages.

### The concept of progressive elaboration takes into account that an initial project plan is usually developed at a very early point in a project lifecycle. Detailed requirements and potential impediments are not yet known at that stage. Therefore, setting up an initial project plan requires assumptions and rough estimates. Over time, initially unknown activities and requirements become clearer and need to be reflected in a refined project plan. This process is called ‘progressive elaboration.

As an example of how progressive elaboration works, consider a software project to build a new website.

* The current sprint includes navigation and basic site structure, the upcoming sprint includes content, and the upcoming release includes “nice-to-have” features such as a customer forum and a product gallery.
* While the team wraps up the current sprint, the product owner gets final feedback on content so that the team can start quickly on the content and finish it in the upcoming sprint.
* If any changes need to be made to navigation and structure, those also go into the upcoming sprint after the current sprint’s review finishes.

### 6 In the past, your team has run projects using the traditional Waterfall model. Suggest the changes you would need to incorporate in terms of roles, values, and philosophy to accommodate agile methodologies.

Here are a few things to try that can help get your team ready for a waterfall to agile transition:

* **Adapt Roles and Responsibilities:**   
  Roles, responsibilities, and styles of working need to change across the company including the product team, business analysts, engineering, and IT. A top-down command-and-control culture should be replaced with horizontal conversations. Management has to help remove impediments, encourage teams, and ensure the business alignment of projects.   
    
  Evangelize Agile culture and introduce teams to agile roles like SCRUM master, Agile coach, Product Owner and Agile development teams. Introduce teams to Agile ceremonies like Sprint Planning, Retrospectives, Daily Standup, etc. Create learning content for them to get familiar with and start adopting these practices.
* **Agile Values:**The whole-team approach involves constant collaboration. The team needs to collaborate right from the beginning: when it comes to defining requirements, the scope of projects, and what is needed to assure the quality of the software. Software quality is a continuous process that requires commitment from everyone involved.   
    
    
  Agile gives everyone a voice and values every person’s opinion. This can be the biggest change for teams where they were used to only the managers speaking while others nod in unison.   
    
  Promote the Agile Values as:  
  + Individuals and interactions over processes and tools.
  + Working software over comprehensive documentation.
  + Customer collaboration over contract negotiation.
  + Responding to change over following a plan.

* **Agile Philosophy:**The Agile philosophy is client-centric and supports ongoing adjustments based on feedback from clients. No matter how clean the code is, the developed product has to meet the client’s needs to maintain its competitive advantage in the market. Agile helps us prioritize client feedback and implement improvements that are essential to boost client experience further.

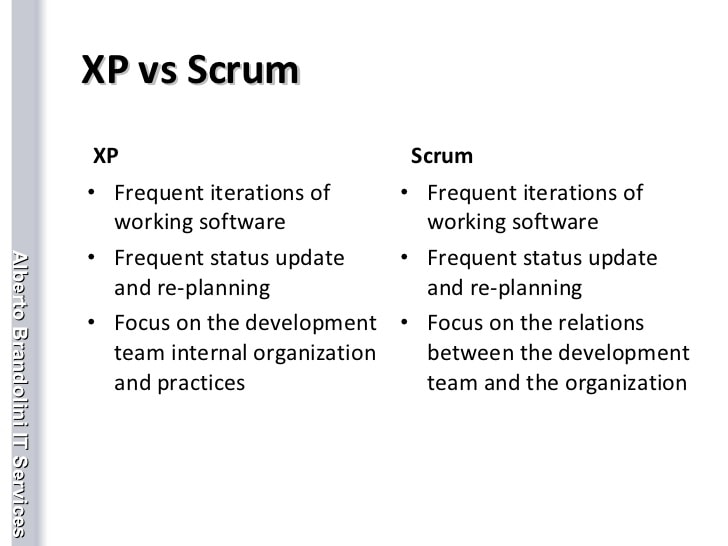
### 7 Briefly explain the poker estimation technique used to estimate requirements. Explain the roles of the scrum master and the product owner in this context. How do they help to resolve a deadlock here?

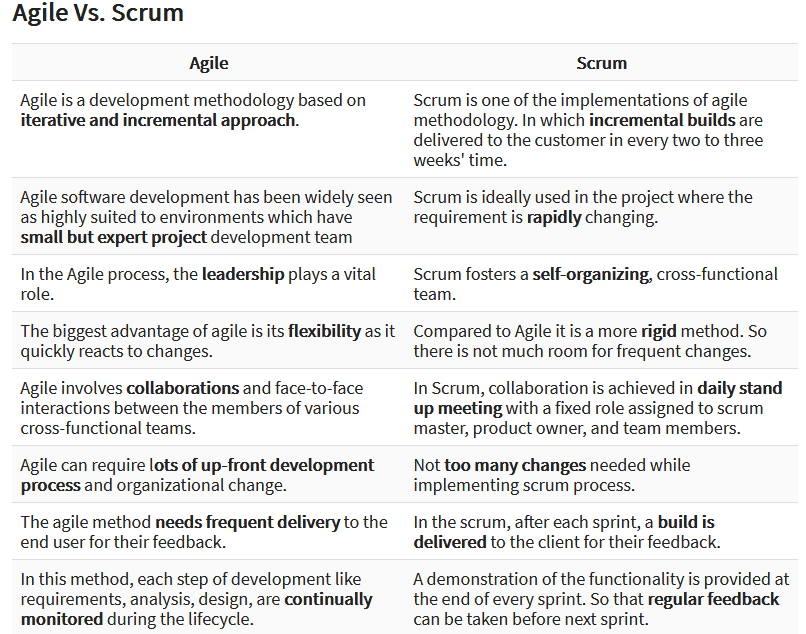
The software project delivery schedule is driven by business needs. For the team to commit to the deadlines, it is important for the team to come together and provide a realistic estimate.

Planning poker (also known as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/) poker) is a consensus-based, gamified technique for estimating, mostly used to estimate effort or relative size of development goals in software development. The team will take an item from the backlog, discuss it briefly, and each member will mentally formulate an estimate. Then everyone holds up a card with the number that reflects their estimate. If everyone agrees, great! If not, take some time to understand the rationale behind different estimates.

1. To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators.  
   For example:  
   “Customer logs in to the reservation system”  
   “Customer enters search criteria for a hotel reservation”
2. Team members of the group make estimates by playing numbered cards face-down to the table without revealing their estimate (Fibonacci values: 1,2,3,5,8,13,20,40)
3. Cards are simultaneously displayed
4. The estimates are then discussed and high and low estimates are explained
5. If there is no convergence, the developingScrum master and team discuss the highs and lows. Scrum master also does conflict management.
6. The Scrum Master can do several tries to get a consensus for the story points by repeating the vote and usually after 2-3 tries this works.
7. Repeat as needed until estimates converge

### 8 Explain three similarities and three dissimilarities between XP and SCRUM.

****

****

### 9 What is an after-party meeting? Why is it crucial to have this meeting? Give at least two reasons. Who are the attendees of this meeting? [3]

After Party meetings are organized to solve problems when the daily scrum is finished. Some scrum teams schedule time for an after-party every day; others only meet as needed.

Two reasons why this is crucial are:

* Scrum has a fixed time constraint of 15 minutes and sometimes there may be issues faced in developing participate regularly, the team needs to come to the Standup Meeting and aims to participate regularly. The Standup Meeting aims discussion. The team should be always on top of the development and issues/fixes to address
* In the case of technical blockers, it would be good to discuss with the expert, that skillset within the team save and saves time to resolve team save. This avoids eating the time of other members who may not need to be a part of these discussions.

The audience for an after-party meeting is generally not more than 3 to 4 people, which includes members of the Scrum team that need to address or consult technical or development issues.

### 10 Assuming you are monitoring an agile project as a mentor. Initially, the project was running fine. However, lately, you observe that the team reports missing functionalities after sprints. Where is the problem potentially? Suggest three strategies to tackle this. [3]

As the Agile Mentor for a project, let's also assume the teams are following all Agile ceremonies and practices. In this scenario, if we notice that there are missing functionalities in the team’s reports, it means that the problems might be :

* Developers will pick up tasks from user stories, discuss impediments and build them into the next incremental release. But, since the functionalities are missing in the reports, it leads to user stories not being defined properly (assuming, developers have not swayed from the user stories and adhered to the requirements), the problem is related to details not incorporated in the user story or improper tracking of requirements.

Three strategies to tackle this are:

1. **Add as many details to the user story as possible.** To shorten the time for development and testing and save money for the client you should provide as many acceptance criteria to the user story as possible. Ideally, if the user story has a prototype attached, later you will be able to easily add new or edit existing user stories based on the feedback from earlier prototypes***.***

*2.* ***Adopt a use case approach and template*.** Write basic examples with required fields for creating a user story or a task in a defect management tool like Jira. Thus, a person who writes requirements won’t forget to fill in any single field.

For example:

* Acceptance criteria
* Scenarios
* Design
* Web/native
* How to demo the user story
* Mobile/tablet/desktop
* All needed hardware/software available for testing
* Platforms
* Affected areas to test
* Where/when needed to be delivered
* PO/BA/Designers POC
* Error handling.

3. **Conduct repeated grooming sessions and issue triage**. If after a grooming session some statements are still unclear, clarify this with the appropriate person, ask stakeholders their opinion on the approach. If the description isn’t detailed enough for anyone in the meeting, groom one more time until everyone is on the same page and understands feature specifics.

**5. Track requirements status and communicate with stakeholders.** If someone from stakeholders or the development team has commented or replied to the question in Jira or another relevant management tool then these clarifications need to be amended or added to acceptance criteria in the user story. The same applies to design-related documents. All mock-ups should be updated in the user story not to confuse developers so that they know exactly how the feature should be designed.

A good practice here would be to invite a designer to the meeting with stakeholders and then link design explorations to the user story. If communication with stakeholders was conducted via mail/Slack/Skype, then all updated and confirmed data need to be copied to the ticket by the person responsible.

### 11 Explain at least two tasks of the scrum master, development team, and the product owner while a sprint is running.

While a sprint is running, the roles of:

* **Scrum Master:**
  + Maximize productivity and handle any blocking issues which require inputs from the Product Owner.
  + Prepares the product backlog for the next sprint.
  + Resolve conflicts by focusing on scrum values of openness, honesty, and respect.
  + Forecast the number of deliverables possible in an iteration based on evidence.
* **Development Team:**
  + Perform the tasks of designing, building, integrating, and testing product backlog items into increments of potentially shippable functionality.
  + Inspect their progress toward the sprint goal and adapt the plan for the current day’s work.
  + In case of roadblocks or spikes, they have to bring it to the notice of the Scrum master and the rest of the team
* **Product Owner:**
  + Responsible for the outcome of the sprint cycle
  + Decides which user stories should be ideally taken up for development purposes based upon their business values.
  + Sharing knowledge, skills, and experience with the team members whenever required, if the team asks for guidance.

### 12 Why do you think the waterfall model is not suited for current software development trends? Give a scenario where it might be better suited than agile methodologies.

A waterfall is a sequential approach, separating a project into different phases. In waterfall projects, requirements are gathered in the analysis phase and are collected and documented in a specification document. After completing the analysis stage, the requirements should remain stable. Once the product is finished, the whole bulk of the value is delivered to the customer.

In reality, most software development projects have to change some of the requirements eventually. You often hear people speaking of scope creep. Instead of embracing change, taking feedback, and identifying opportunities to add value to a product, project teams will usually try to avoid change and settle with a mediocre product because going back to the analysis phase usually means delaying the whole project. Hence modern projects and current development trends do not prefer waterfall.

A scenario where Waterfall is better suited than Agile would be:

In a large-scale manufacturing operation where all of the parts need to be identified, designed, and moved into manufacturing at roughly the same time (think of something like airplanes, cars, and other massive industrial systems). In these cases, there is no room for alterations in the overall design, or even in many of the subsystems, while the process is in place, and the specific needs and problems that are being solved by the system are clear, well-known, and pretty much unchanging.

### 13 Write a User Story for Agile Conference in the following format:

The format for this exercise is: As a <type of user>, I want <to perform some task> so that I can <achieve some goal/benefit/value>                  [2]

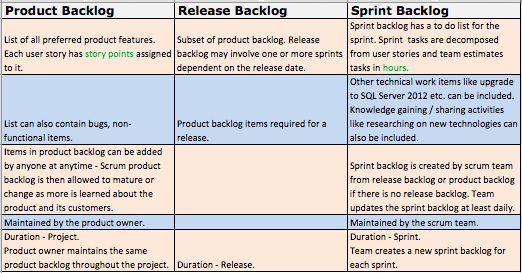
As an attendee to the Agile Conference, I want to be able to see the list of speakers in this conference and their session timings, so that I can structure and plan my day to select what sessions I want to attend during the conference.

### 14 Match the following:

**Ans:**

1. Emotional Intelligence - D
2. Collaboration - C
3. Adaptive leadership - A
4. Negotiation - B
5. Conflict Resolution - E.
6. Servant Leadership - F

### 15 Difference between Product Backlog and Sprint Backlog



### 16 Questions during daily standup meetings: Which of the following is valid during daily standup meetings and why?

1. Our new contractor can’t start because no one is here to sign the contract.

2. The department manager has asked me to work for another team for the remaining week.

3. I tested story number 10 yesterday.

4. I still haven’t received the software I ordered a month ago.

5. You discuss the football match since it involves a bet between several team members.

6. I’m struggling to learn C++ and would like to pair with someone on it.

I can’t get the product owner to show up or call me back.

8. You explain why a team member should be removed from the team.

Ans:

Questions valid are, 2,3,4,6 (if project has C++ dependencies, otherwise not) ,7

### 17 Your team is starting to use story point estimation for the first time. How do they know how many stories point to assign to a story?

Teams assign story points relative to work complexity, the amount of work, and risk or uncertainty. Values are assigned to more effectively break down work into smaller pieces, so they can address uncertainty.

Over time, this helps teams understand how much they can achieve in a period and builds consensus and commitment to the solution.

If it is the first sprint, collectively judge how many stories they feel they can finish in the first sprint. The story points will not become valuable until the team executes a few sprints anyway, and a velocity metric can be identified. Then the team can look at their velocity as a guide for how much work they can comfortably accept each sprint, and the PO can look at the velocity metric to estimate how much of the product backlog can get done by certain dates.

### 18 To develop team collaboration and commitment it may be necessary to reduce distractions to establish a predictable outcome and optimize the value delivered. The following table contains the roles found in Scrum. Explain what each role should do to reduce distractions.                                                     [3]

|  |  |
| --- | --- |
| **Role** | **How to Reduce Distractions?** |
| The Team | 1. Discuss blockers 2. Organize After Party meetings to discuss and address issues immediately 3. Update and document tasks in their sprint tracker tool 4. using time management tools to plan their day effectively |
| The Scrum Master | 1. Hold people accountable 2. Incorporate Motivation 3. Ensure requirements and user stories are clear and structured |
| The Product Owner | 1. Keep track of project progress with Burndown charts 2. Constantly clarify doubts when required and convey information about changes to requirements on time 3. Connect with the teams more frequently |

### 

### 19 ABC Corporation has received the contracts from the customer to execute the following types of software projects.

**Project-A: A mission-critical software development project with adherence to external regulatory compliance. In this project, the customer has committed upfront to provide detailed product requirements. These requirements are reviewed and approved. The customer does not expect any major change in scope during the execution of the project.**

**Project-B:  An emerging e-commerce development project. In this project, the customer has committed upfront to provide high-level, get, he Standup Meeting aims requirements. These requirements are evolving and subject to change during the project.**

**In both the projects, the customer is expecting the delivery of the working product only at the end of the project and has agreed to collaborate, review and provide feedback whenever required. The expected duration of these projects is around 6 months and you may assume adequate resources are available for both projects.**

**As an Agile consultant, what is the project life cycle method best suitable for each type of project and explain the rationale behind choosing the life cycle method? What are the development phases involved in each life cycle method?[4]**

Project A:

In this case, where the customer doesn’t expect any major change in the scope during the execution of the project, the Waterfall model can be used as Waterfall methodology depending on the belief that all project requirements can be gathered and understood up front.

Phases of the system generally include:

* Requirement gathering and documentation
* System design
* Implementation
* Testing
* Delivery/deployment
* Maintenance

Project B:

In this case, where the requirements are evolving and are subject to change, the Agile Method is recommended to be used as it helps balance the need for change with the need for a stable development environment.

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders.

The Agile software development life cycle is the structured series of stages that a product goes through as it moves from beginning to end.

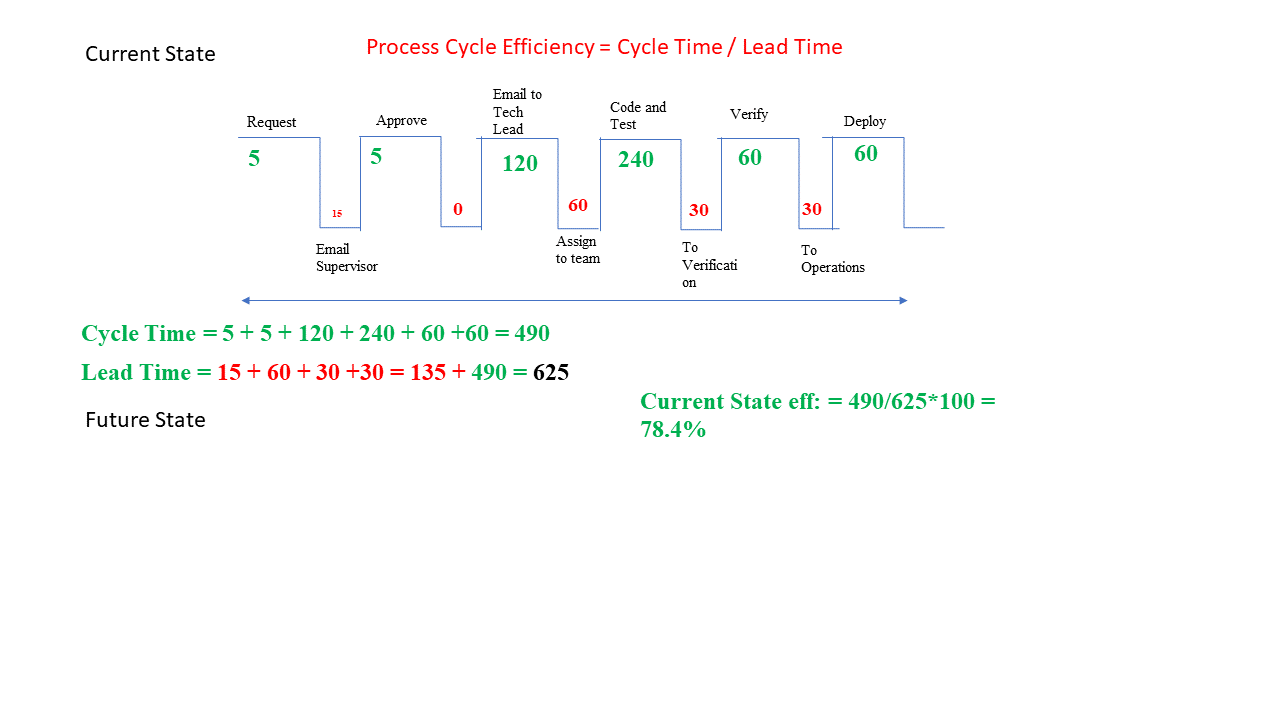
It contains six phases:

* concept
* inception
* iteration
* release
* maintenance
* retirement

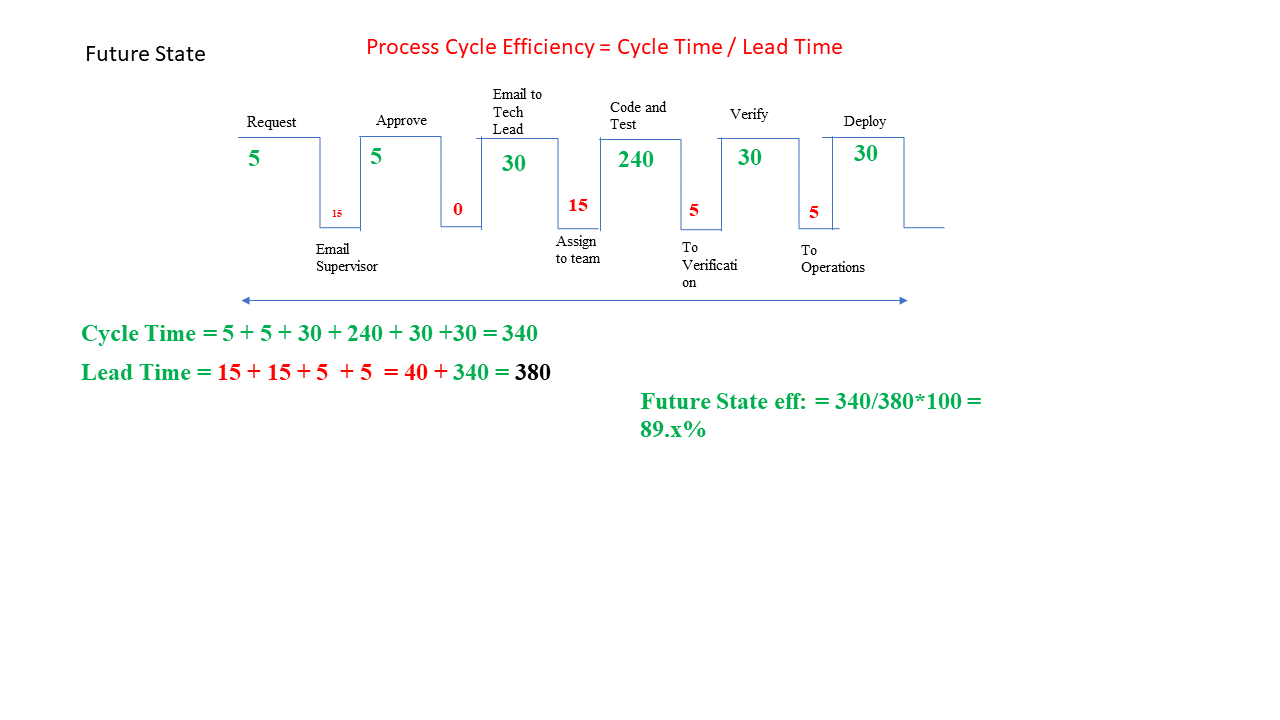
### 20. The process found in the following table has an efficiency of 76% (490/640) with 490 minutes of value and 150 minutes of waste.  Draw the Value Stream Map of the current state of the process? How would you improve on this process?[4]

|  |  |  |
| --- | --- | --- |
| **Activities** | **Value** | **Waste** |
| Request | 5 minutes |  |
| E-mail supervisor |  | 15 minutes |
| Approve | 5 minutes |  |
| E-mail to tech-lead | 2 hours |  |
| Assign team |  | 1 hour |
| Code and test | 4 hours |  |
| To verification |  | 30 minutes |
| Verify | 1 hour |  |
| To operations |  | 30 minutes |
| Deploy | 1 hour |  |

Current State:



Future State:



### 21 Agile projects deliver high-quality products, which to some degree, are managed by effective problem detection and resolution. Which tools and techniques are applied for Agile teams to deliver high quality and briefly explain the techniques?[4]

Agile methods and tools lend themselves most appropriately to systems and projects in which accurate estimates, stable plans, and predictions are often difficult to attain in the early project stages. Agile development favors an adaptive, iterative, and evolutionary development approach. Using agile tools and techniques can help to:

* Self-organize and plan
* Communicate (within the team and with the rest of your organization)
* Continuously improve the way you work
* Get support from management

Agile Tools and Techniques

The common agile tools and techniques that teams use can be used depending on the project and team. Agile teams may also adopt unique combinations of techniques to support their framework and methodology. Examples include:

* Collocation
* Dedicated teams
* Relative estimating
* User stories
* Velocity
* Burndown charts
* Definition of done

A few techniques typically used on agile projects that directly contribute to accelerating the time to delivery and the increased quality of the product being delivered, which include:

* Set priorities.

A product backlog is a list of prioritized tasks maintained by a product owner.

* Maintain small release cycles.

The product should be released in increments every 2-4 weeks, with stakeholders giving feedback before proceeding.

* Use pair programming.

Two programmers work side-by-side at a single computer. This technique results in an identical degree of productivity to separate programming but delivers higher quality.

* Refactor.

Rework code regularly to achieve the same result with greater efficiency and clarity.

* Use test-driven development.

Code the unit test first to keep the project on task throughout. Test-driven development as an Agile best practice also produces greater employee engagement, since it transforms testing from a boring grind to a coding challenge.

### 22 You are taking on the role of Scrum master of an Agile team. Several team members have been removed because they didn’t deliver and had a bad attitude. Now team members are not showing up for meetings, the team is running low in each sprint, and the product owner complains that working with them is a hassle and the quality of their work is low. What could be the possible problem with the team and what can you do?[2]

Talk to the employees and provide them with examples of bad behavior – One way to make feedback specific is to highlight past examples of the employee’s poor attitude.

Give actionable advice – After you provide examples of bad behavior, clearly let the employee know how they should have behaved so they know what is expected of them going forward.

### 23 Your customer has the following customer satisfaction issues.

**a. Product requirements misunderstood by the development team**

**b. The customer can’t request changes without additional cost and time.**

**c. The product wasn’t delivered when the customer needed it.**

**Explain how by adapting Agile approaches, these issues can be addressed more effectively? [3]**

1. When dealing with complex requirements, the development team along with ScrumMaster needs to plan and design the solution as best as possible. That means breaking up complex requirements into smaller stories and iterating over time.

If the team sees any impediments or if ScrumMaster notices anything that will be a roadblock in the future – all those issues should be raised ahead of time and a plan should be put in place. While you cannot account for all issues, it is important to know that every change made to the app during iterations has a cost. Sometimes developers change really big features late in the project. And while the developers may understand the implications of this change, the end-users expect that since it’s agile, things will just be okay and fix themselves. However, the only way for a project to succeed is to add other iterations and extend the deadline.

1. Agile development is a process in which the requirements are expected to change and evolve. By using technical practices like Continuous Delivery and Test Driven Development they reduce the cost and time involved with testing and deployment. Doing this makes it possible to test and deploy far more frequently and an agile process reduces the agility and cost of change when compared to other models because software is released in increments and change can be better controlled within an increment.
2. You'll have to reprioritize the original requirements so they can be completed within the approved time frame. To do that, you must identify which components are most critical to delivering. These are the non-negotiable must-have factors that the stakeholder can't live without.

A helpful technique in this scenario is timeboxing, which is based on the premise that it's better to have a working system with limited functionality than waiting for more time to have a complete system. With this technique, you can guarantee the delivery of the most important requirements on specific dates, with other requirements scheduled for release on successive dates or phrases.

### 24 Different project life cycle methodology, identifying a suitable life cycle for a project?

The Project Management Lifecycle is used to describe a deliberate, structured, and methodical process for Project Management. The Project Management Life Cycle has five phases: Request, Initiation, Planning, Execution & Control, and Closure.

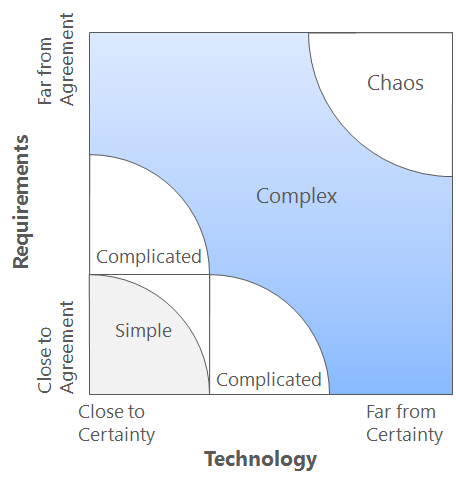
* Requesting
  + Delivers a project request
  + Captures specific project information that will aid in project evaluation
  + Approves the project request to advance to the Initiation Phase (or defers the request
  + Completed and approved project requests constitute the Project Portfolio
* Initiation
  + Formalizes the existence of the project
  + Defines the preliminary project cost, scope, roles, and timeline
  + Approves the project to advance to the Planning Phase
  + Delivers a Project Charter
* Planning
  + Defines the detailed project schedule, budget, resources, allocations, and timeline.
  + Provides the baseline to control and manage the project
  + Identifies the project risks
  + Approves the project to begin work
  + Delivers a detailed Project Plan.
* Execution & Control
  + Creates and/or delivers the end product or service
  + Executes the tasks in the project plan (schedule)
  + Manages the project issues & risk
  + Relies on the plans from the Planning Phase to control the project.
  + Delivers regular updates to stakeholders detailing progress
* Closing
  + Concludes all project tasks/activities
  + Administratively closes the project
  + Gathers and catalogs Lessons Learned
  + Turns the delivered product or service over to the customer or a support group
  + Assesses project outcomes and team performance
  + Delivers a Project Closeout Report

This methodology contains the templates for the various project management activities undertaken to deliver successful projects. Each project phase addresses a specific aspect of the process of managing a project from initiation through close. Although these phases are described sequentially, in practice many of these phases may overlap or be applied concurrently during the lifetime of a project.

### 25 Complexity models - Stacey’s Complexity Model, Cynefin framework

#### Stacey’s Complexity Model

A very powerful concept for understanding uncertainty in Agile Projects is the “Stacey Complexity Model”. There are two dimensions of uncertainty in this model:



**Requirements Uncertainty**

One dimension is requirements uncertainty – How well understood are the goals and requirements for the project and how certain are the customers that they know what they want?

**Technology Uncertainty**

The other dimension is technology uncertainty – How well understood is the technology solution to the problem and what is the level of risk associated with the technology solution?

This is a very important concept because the ability to handle uncertainty is so important in today’s most critical projects and heavily plan-driven projects are not well-designed to deal with high levels of uncertainty.

**Management of Uncertainty**

What typically happens in a plan-driven project is the project manager tries to reduce the level of uncertainty to an acceptable level before starting the project by:

Trying to resolve any uncertainties in the requirements as much as possible before the project starts, and Trying to eliminate as much technology risk as possible.

This often results in using tried-and-proven technology and doesn’t push the envelope too far in terms of going into areas of new and undefined user requirements. The downside of that, of course, is that the technology approach may wind up being obsolete within a relatively short amount of time after it is released and it may also result in a very mediocre user experience with the solution.

What Does “Managing Uncertainty” Mean?

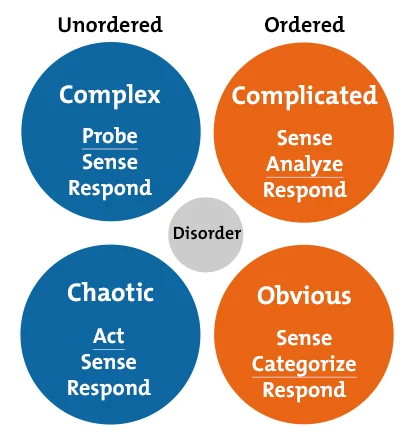
To some people, uncertainty is like cancer that attacks a project and can cause it to fail. Conventional project management thinking has reinforced this approach to reduce the risk and uncertainty in a project as much as possible

I don’t see it that way. Uncertainty can also represent an opportunity to go beyond what is expected and the value a project produces is many times directly related to the level of risk and uncertainty in the project

If you force a project to fit into a plan-driven model by reducing the risk and uncertainty, you may be maximizing the predictability of the project to meet cost and schedule goals but minimizing the value that the project produces.

#### Cynefin framework

A Cynefin framework is a problem-solving tool that helps you put situations into five "domains" defined by cause-and-effect relationships. This helps you assess your situation more accurately and respond appropriately.



You can use the Cynefin framework in a variety of situations to categorize a problem or decision and respond accordingly. For example, it is useful in product development, marketing, and organizational strategy. It can also help you make better decisions in a crisis or emergency.

It helps you avoid using the same management style or decision-making approach in all situations – a mistake that can be costly to your team or organization– by encouraging you to be flexible and adaptable when making decisions and to adjust your management style to fit your circumstances.

### 26 Issues with Waterfall model

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty are high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

### 27 Agile Manifesto and principles.

[3 - Agile principles and manifesto](#_heading=h.d28derz4bl3s)

### 28 Agile methodologies: Scrum, XP, Lean, Value stream mapping, and Kanban Roles, events, technical practices, and principles.

[4.1: Overview of SCRUM Methodology](#_heading=h.johac0u8t4wg)

[4.2: Extreme Programming (XP)](#_heading=h.qa2d6xy6s7qh)

#### Lean

Lean Software Development (LSD) is an agile framework that is used to streamline & optimize the software development process. It may also be referred to as Minimum Viable Product (MVP) strategy as these ways of thinking are very much alike since both intend to speed up development by focusing on new deliverables.

**Advantages of LSD**

LSD has proved to improve software development in the following ways :

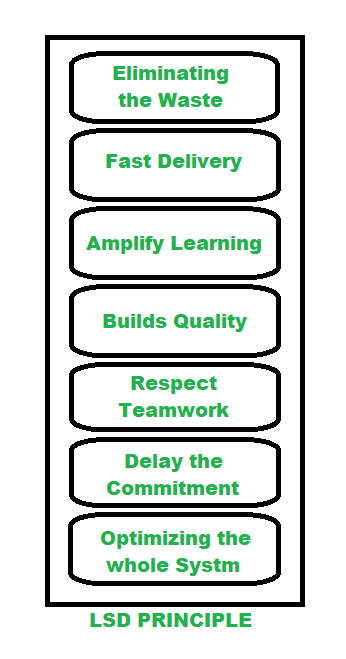
* LSD removes the unnecessary process stages when designing software so that it acts as a time saver and simplifies the development process.
* With a focus on MVP, Lean Software Development prioritizes essential functions so this removes the risk of spending time on valueless builds.
* It increases the involvement power of your team as more and more members participate due to which the overall workflow becomes optimized and losses get reduced.

**Key Principles of Lean Software Development**

7 established lean principles come with a set of tactics, practices, and processes that builds more efficient software products :

* Eliminating the waste
* Fast Delivery
* Amplify Learning
* Builds Quality
* Respect Teamwork
* Delay the commitment
* Optimizing the whole system

**Principles of LSD**



**Eliminating the Waste**: To identify and eliminate wastes e.g. unnecessary code, delay in processes, inefficient communication, the issue with quality, data duplication, more tasks in the log than completed, etc. regular meetings are held by Project Managers. This allows team members to point out faults and suggest changes in the next turn.

**Fast Delivery**: Previously long time planning used to be the key success in business, but over time it is found that engineers spend too much time on building complex systems with unwanted features. So they came up with an MVP strategy which resulted in the building products quickly that included a little functionality and launching the product to market and seeing the reaction. Such an approach allows them to enhance the product based on customer feedback.

**Amplify Learning**: Learning is improved through ample code reviewing, meeting that is cross-team applicable. It is also ensured that particular knowledge isn’t accumulated by one engineer who’s writing a particular piece of code so paired programming is used.

**Builds Quality**: LSD is all about preventing waste, keeping an eye on not sacrificing quality. Developers often apply test-driven programming to examine the code before it is written. The quality can also be gained to get constant feedback from team members and project managers.

**Respect Teamwork**: LSD focuses on empowering team members, rather than controlling them. Setting up a collaborative atmosphere, keeping perfect balance when there are short deadlines and immense workload. This method becomes very much important when new members join a well-established team.

**Delay the Commitment**: In traditional project management it often happens when you make your application and it turns out to be completely unfit for the market. LSD method recognizes this threat and makes room for improvement by postponing irreversible decisions until all experiment is done. This methodology always constructs software as flexible, so the new knowledge is available and engineers can make improvements.

**Optimizing the whole system**: lean’s principle allows managers to break an issue into small constituent parts to optimize the team’s workflow, create unity among members, and inspire a sense of shared responsibility which results in enhancing the team performance.

**Weakness in LSD**

* Make it scalable as other frameworks since it strongly depends on the team involved.
* It is hard to keep pace so it is not easy for developers to work with team members as conflict may occur between them.
* It leads to a difficult decision-making process as customers must set, their requirements for the development not to be interrupted.
* Lean Software Development is one of the proactive approaches that drives your body through productivity and cleanliness. It closely connects to Agile methodology, knowledge-sharing experience, and fast product delivery. All processes and stages of development are accurately built to deliver the end product at minimum cost promptly.

#### Value stream mapping

*Value stream mapping is a Lean management method that allows you to visualize, analyze and improve all the steps in a product delivery process.*

The Value stream mapping process allows you to create a detailed visualization of all steps in your work process. It is a representation of the flow of goods from the supplier to the customer throughout your organization.

For example, the value a software company delivers to its customers is software solutions and all features inside.

A value stream map displays all the important steps of your work process necessary to deliver value from start to finish. It allows you to visualize every task that your team works on and provides single glance status reports about each assignment's progress.

**Purpose and Benefits**

The primary purpose of creating a value stream map is to show you where you can improve your process by visualizing both its value-adding and wasteful steps.

You just have to display every vital step of your workflow and evaluate how it brings value to your customer. This allows you to analyze your process in-depth and provides you with precise insights into where you should make changes to improve the way you work.

To summarize the benefits of value stream mapping:

* With the help of VSM, you can identify wasteful activities.
* VSM provides a clear view of the work process - where value-adding and non-value-adding stages form. A good practice is also to visualize how long it takes for work items to go through them.
* VSM highlights the current workflow and brings the focus on future improvement.

#### 

#### 

#### Kanban Roles

*“Respect the current process, roles, responsibilities, and titles.”*

This principle explains how Kanban does not come with an extensive list of roles or titles to fill.

Teams can start with whatever roles they have and establish new ones that will make sense for them later on. “Start with what you’re doing now,” as we say in Kanban. After years of practice and use in the workplace, Kanban has evolved. There is a clear emergence of two Kanban roles – the Service Delivery Manager and the Service Request Manager.

**Service Delivery Manager**

The Service Delivery Manager (SDM) is responsible for ensuring a smooth flow of work. This kanban role is also known as Delivery Manager, Flow Manager or FlowMaster. It first emerged in 2005 when Microsoft started using Kanban. While we do see organizations establishing SDM as a distinct job position today, its first implementation wasn’t the case. In its early conception, a project manager performed the Service Delivery Manager role. The Project Manager role was then elevated to perform the necessary service delivery functions expected of them.

Service Delivery Managers have the following responsibilities:

* MakeThe Standup Meeting aims sure that work items flow through the Kanban System
* Facilitate continuous improvement initiatives and process reviews
* Facilitate Kanban meetings or the 7 Kanban cadences if the team adheres to it
* Ensure that processes are working towards delivering customer satisfaction

**Service Request Manager**

The SRM initiates the discussions that lead to decision-making around these service requests. Imagine the SRM facilitating a discovery session with stakeholders, account managers, and product managers. Add in a kanban board filled with ideas and suggestions for their product’s next release. They would go over each one and facilitate the exchange of opinions and ideas from everyone present in the room. The SRM doesn’t call the shots, but they do maintain the policies that govern the selection and review process for these service requests.

The Service Request Manager is often compared to and regarded as a similar role to the Product Owner in Scrum. While there may be some similarities, the intention for the roles is completely different. While the SRM is an advocate for the needs of a product or service’s end-customers, it is not meant to perform decision-making responsibilities around what gets into the product and what doesn’t.

#### Kanban Events

The meetings lead back to the same goal – faster delivery, more efficient flow, and improved client satisfaction. You are free to adapt them according to the needs of your organization. Certain events can also be used to trigger meetings – for example, risk reviews are generally conducted monthly but could be triggered by a critical glitch that made it through to production.

* Standup Meeting

Suggested Frequency: Daily

Suggested Length: 15 minutes

The key feature of the Standup Meeting is that it is quick and efficient – it is traditionally held standing, so nobody gets too comfortable. The aim of the Standup Meeting is to answer the following three questions:

* What’s impeding us?
* How is work flowing?
* What can we improve?
* Replenishment Meeting

Suggested Frequency: Weekly

Suggested Length: 30 minutes

To keep a steady stream of tasks moving across the Kanban board, the number of tasks in the backlog must be selected. This takes place during the Replenishment Meeting. The Kanban Method suggests an approach to backlog management that eliminates the need for manual backlog reordering. We recommend 30 minutes for a Replenishment meeting, however, the frequency will vary according to team needs. A fast-paced workflow with many small tasks can require weekly replenishment meetings, while a slower workflow of large, detailed tasks could only need replenishment monthly.

* Service Delivery Review

Suggested Frequency: Bi-Weekly

Suggested Length: 30 minutes

All the efficiency in the world is wasted if the most important stakeholder – the client – is not satisfied. The Service Delivery Review aims to look at how well the client is being served by the team’s output. An additional benefit of this meeting is cultivating trust with your customer through acting transparently and engaging directly with their concerns.

This meeting should involve the customer (or its representatives), the service delivery manager, and representatives from the delivery team. You might find other stakeholders should be involved depending on the needs of your project. Kanban is a data-driven method that relies upon metrics – consider how client satisfaction criteria can be assessed objectively. Some targets that can be set include desired lead/cycle times, lead time consistency, and overall delivery rates.

* Delivery Planning Meeting

Suggested Frequency: Per delivery cadence (variable)

Suggested Length: 1-2 hours

Work can not always be delivered to clients on the day it is finished – some release dates are inevitably fixed. In a Delivery Planning meeting, the team can predict what needs to be ready for release as well as what other work items are due to be finished. The cumulative flow diagram, throughput histogram, and cycle time scatterplot can be used to make data-driven decisions.

This Kanban meeting should also take into account any hand-off requirements or training activities needed for the client. Ensuring smooth transfers of work goes a long way towards eliminating inefficiencies and keeping customer satisfaction high. Be aware that committing work items to fixed delivery dates should change their Class of Service from standard to fixed dates.

* Risk Review

Suggested Frequency: Monthly

Suggested Length: 1-2 hours

In a self-explanatory Kanban meeting, the Risk Review examines factors that put work delivery at risk. In this meeting, blockers and backlogs should be examined to predict future risks to delivery. The causes of past failures should be assessed and their causes mitigated or resolved. Anyone familiar with current and recent blockers should participate – this will change from month to month, making this the meeting with the most variation in participants.

* Operations Review

Suggested Frequency: Monthly

Suggested Length: 2 hours

The Operations Review takes a holistic view of all the different interconnecting internal teams and systems. Even if individual teams have high efficiency, the whole organization can be held back by one improperly functioning part or hand-off inefficiencies.

This Kanban meeting involves managers from different divisions, departments, and systems looking for ways to improve the efficiency of the whole. Particular attention should be paid to interdependencies between Kanban systems that can have a ripple effect on overall delivery times. The Operations Review is also an ideal time to spot areas of underused capacity throughout the organization that can be used to shorten lead times.

* Strategy Review

Suggested Frequency: Quarterly

Suggested Length: Half-day

The Strategy Review takes a big-picture look at the whole operation – a perfectly efficient vehicle is of little use if driving in completely the wrong direction. This Kanban meeting takes into account the larger market landscape, examines new changes, and compares delivery speeds with the rate of market changes. The wider strategic goals and direction can be used to set a Kanban roadmap. Translating direction into monthly, weekly and daily goals can then take place during Delivery Planning Meetings and Replenishment Meetings.

The aim is to identify potential large-scale problems and course-correct or optimize team operations where necessary. Ideal participants for the Strategy Review include senior executives, portfolio/product owners, senior team members from customer-facing departments such as sales and marketing.

### 29 Task boards. User stories, Estimation, velocity calculation Release planning

#### Task Boards

The Task Board is perhaps the single most useful, and arguably most important, device that can be used on Agile projects, often described as an ‘information radiator’ because it gives out the information to everyone from a central location. A Task Board is the focal point of any Agile project and serves as a good place at which to hold the stand-up meeting or Scrum.

Typically, a Task Board displays only information pertinent to the current sprint and will be cleared off before the next sprint begins. There can be exceptions such as reminders, technical notes, or other data that needs to be easily available and persists across iterations, although such information is not strictly part of the Task Board.

**Task Board Placement**

It is important that the Task Board be somewhere visible to as many team members as possible and always accessible to all; usually a large wall or window (be creative). Manager’s offices and conference rooms are not good locations as they are not always accessible. It is important that every team member feels they share ownership of the Task Board; it is representative of the teamwork necessary to succeed and shows that each individual is responsible for that success (or even failure.)

**Creating a Task Board**

The surface area doesn’t matter provided it is relatively large, that horizontal and vertical lines can be placed on it, and that notes of some kind can be attached. Almost all surfaces are suitable for some kind of tape to create the lines, but the method of displaying notes or cards can vary by surface.

* Corkboards with push-pins,
* Whiteboards, windows, large pieces of paper, or large plastic wall calendars, all with reusable sticky notes,
* Metal surfaces with paper and magnets, or
* Any basic wall surface with removable, non-marking labels.

The information is displayed in a 2-dimensional matrix with rows representing User Stories and columns representing various status values, which should be, at a minimum:

* tasks to do,
* tasks in progress, and
* tasks done.

**Software Task Boards**

With a Task Board being a 2-dimensional matrix, it could be represented in a spreadsheet, but moving tasks around in a spreadsheet is neither sufficiently quick nor simple. There are software tools that help manage tasks or represent the Task Board and these can work well, especially for teams that are not co-located.

One of the benefits of a software-based task board is that you can rearrange it to show different views in real-time. For example, in Scrum projects, you often want to show the task board by user story or by a person.

But it is not always the best way to start. A team with less Agile experience would do well to use a physical Task Board in parallel with any software tool to get the best from both worlds.

#### User Story

Requirements always change as teams and customers learn more about the system as the project progresses. It's not exactly realistic to expect project teams to work off a static requirements list and then deliver functional software months later.

With a user story approach, we replace a big upfront design with a "just enough" approach. User stories reduce the time spent on writing exhaustive documentation by emphasizing customer-centric conversations. Consequently, user stories allow teams to deliver quality software more quickly, which is what customers prefer. There are quite a few benefits for adopting a user story approach in agile development such as:

* The simple and consistent format saves time when capturing and prioritizing requirements while remaining versatile enough to be used on large and small features alike.
* Keep yourself expressing business value by delivering a product that the client needs
* Avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution.
* Avoid the appearance of false completeness and clarity
* Get to small enough chunks that invite negotiation and movement in the backlog
* Leave the technical functions to the architect, developers, testers, and so on
* Basic Concepts of User Story
* A user story is a lightweight method for quickly capturing the "who", "what" and "why" of a product requirement. In simple terms, user stories are stated ideas of requirements that express what users need. User stories are brief, with each element often containing fewer than 10 or 15 words each. User stories are "to-do" lists that help you determine the steps along the project's path. They help ensure that your process, as well as the resulting product, will meet your requirements.

A user story is defined incrementally, in three stages:

* A brief description of the need
* The conversations that happen during backlog grooming and iteration planning to solidify the details
* The tests that confirm the story's satisfactory completion

And these, although, are known as the 3C's - Card, Conversation, and Confirmation. We will talk more about this later on in this user story guide.

A good user story should be - INVEST:

* **Independent**: Should be self-contained in a way that allows them to be released without depending on one another.
* **Negotiable**: Only capture the essence of the user's need, leaving room for conversation. A user story should not be written like contract.
* **Valuable**: Delivers value to end-user.
* **Estimable**: User stories have to able to be estimated so they can be properly prioritized and fit into sprints.
* **Small**: A user story is a small chunk of work that allows it to be completed in about 3 to 4 days.
* **Testable**: A user story has to be confirmed via pre-written acceptance criteria.

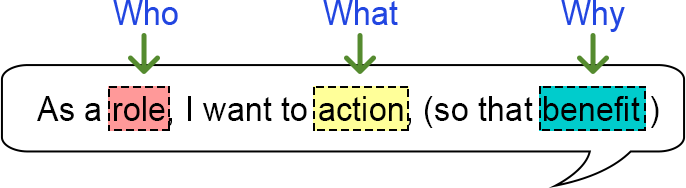
**How to Write User Stories?**

When getting started with writing user stories, a template can help ensure that you don't inadvertently start writing technical tasks:

**User Story Template**

User stories only capture the essential elements of a requirement:

* Who is it for?
* What does it expect from the system?
* Why is it important (optional?)?
* Here is a simple format of user story used by 70% of practitioners:



**Role** - The user should be an actual human who interacts with the system.

* Be as specific as possible
* The development team is NOT a user
* Action - The behavior of the system should be written as an action.

**Benefits** - The benefit should be a real-world result that is non-functional or external to the system.

* Many stories may share the same benefit statement.
* The benefit may be for other users or customers, not just for the user in the story.

**User Story Example**

As a [customer], I want [shopping cart feature] so that [I can easily purchase items online].

#### 

#### 

#### 

#### Velocity-based Sprint Planning

The most important measure that an Agile team will use in planning is its “Velocity”. It is the amount of work finished by the team in each sprint. It helps the team to identify how much progress they can aim to make in a given sprint. Velocity is calculated by adding all the story points given to each user story that is completed by the end of the sprint. It measures output, but not the outcome.

The steps involved in Velocity-based Sprint Planning are as follows:

* Calculate the team’s average velocity (from the last 3 Sprints)
* Select the items from the product backlog equal to the average velocity
* Verify whether the tasks associated with the selected user stories are appropriate for the particular sprint
* Estimate the tasks and check if the total work is consistent with past sprints

**Why do we need velocity-based Sprint Planning?**

During sprint planning, the velocity of a team is served as an input to the next sprint. They use this velocity data, evaluate and enhance its use in the scrum to deliver customer value. By evaluating its velocity for the past several sprints, the team can gain knowledge on how the change in the particular process can affect the delivery of measurable customer value.

Moreover, we can:

* Estimate how much amount can be delivered by a particular date
* Estimate date for a committed amount of work to be delivered
* Understand our goals while fixing the amount of work we will commit for a sprint

**Who all participate in the velocity-based sprint planning?**

A velocity-driven sprint planning meeting involves the Scrum Master, Product Owner, and all the development team members. The Product Owner presents the highest-priority product backlog items in the meeting and introduces those top-priority items to the team.

The development team matches their forecasts with actual deliverables, estimate accordingly from current sprint to release. And the team’s velocity is the most appropriate measure used during forecasting.

**How to calculate Sprint Velocity?**

Let’s assume that the team is doing a one-week sprint to calculate velocity.

**Calculating the velocity of sprint 1**

* Assume that the team has committed to 5 user stories
* And each user story= 8 story points
* Then the total story points in sprint 1= 40 story points

Assume that the team has completed 3 user stories out of 5 by the end of sprint 1, then

* Total user stories completed= 3
* Total story points completed= 24 (total no. completed user stories x story points=3 x 8)

**Calculating the velocity of sprint 2**

* Assume that the team has committed to 7 user stories
* And each user story= 8 story points
* Then the total story points in sprint 2= 56 story points

Assume that the team has completed 4 user stories out of 7 by the end of sprint 2, then

* Total user stories completed= 4
* Total story points completed= 32 (total no. completed user stories x story points= 4 x 8)

**Calculating the velocity of sprint 3**

* Assume that the team has committed to 9 user stories
* And each user story= 8 story points
* Then the total story points in sprint 2= 72 story points

Assume that the team has completed 5 user stories out of 9 by the end of sprint 3, then

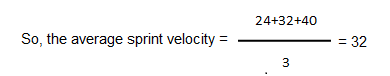
* Total user stories completed= 5
* Total story points completed= 40 (total no. completed user stories x story points= 5 x 8)

According to a ResearchGate study, it was proved that velocity always fluctuates in the first few iterations from sprint to sprint. This is mainly because a new team takes time to get the flow. This also may happen because of some changes happening within a team. The report also demonstrated that in most cases, velocity is likely to stabilize after completing at least three iterations. So, it is good to analyze the completed story points after completing a minimum of 3-5 sprints.

**Calculating average sprint velocity**

We now know our previous sprint velocities which are given below:

* Sprint 1: 24
* Sprint 2: 32



This is our average velocity of the past three sprints, which serves as a reference in understanding how the team is completing the user stories in a sprint. This will give you more clarity and help in planning your future sprints perfectly. When you are planning a sprint, velocity will give you the reference as to how many user stories can be done in a sprint.

**Note:** The total number of user stories committed during a sprint should not exceed the average velocity of the past sprints.

**Reasons for fluctuations in velocity:**

Velocity can fluctuate based on the following variables in a given project:

* Project complexity
* Team size
* Uniformity in team membership
* Team ability to concentrate on Scrum stories and activities
* System outages
* Lack of Product Owner engagement
* Unexpected absences in the team

**Impact of velocity-based sprint planning**

Release planning is not possible without velocity. A Product Owner, by estimating the velocity, can predict the number of sprints required by the team to achieve a desired level of functionality that can then be dispatched. Therefore, the Product Owner can fix a specific date for release depending on the sprint length.