What is Software Development?

Software development is the process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components. Software development involves writing and maintaining the source code, but in a broader sense, it includes all processes from the conception of the desired software through to the final manifestation of the software, typically in a planned and structured process.[1] Software development also includes research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products.

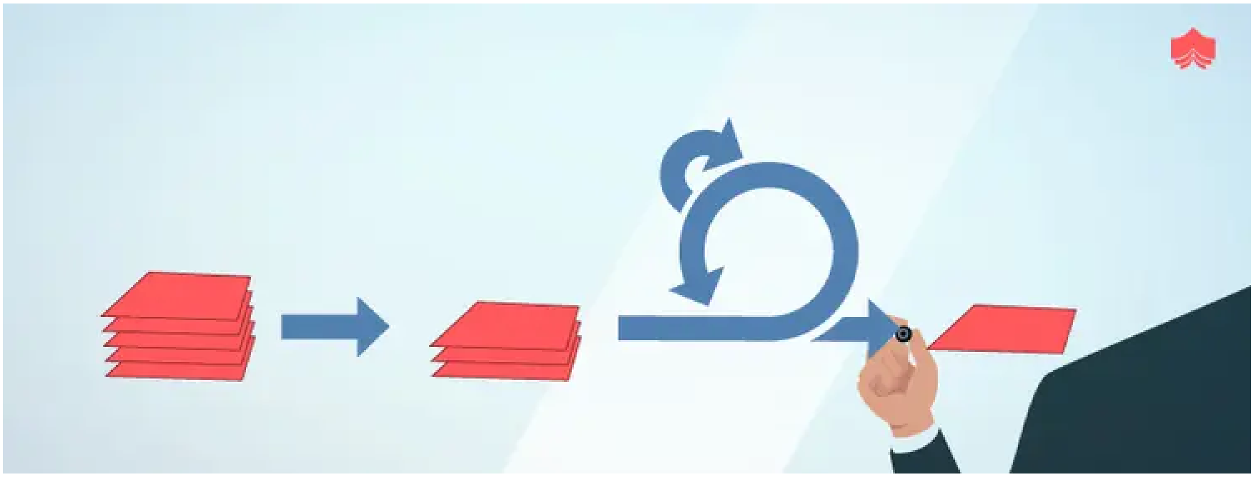
What is Agile Methodology?

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.

It's a process for managing a project that involves constant collaboration and working in iterations. Agile project management works off the basis that a project can be continuously improved upon throughout its life cycle, with changes being made quickly and responsively. Agile is one of the most popular approaches to project management due to its flexibility, adaptability to change, and high level of customer input.

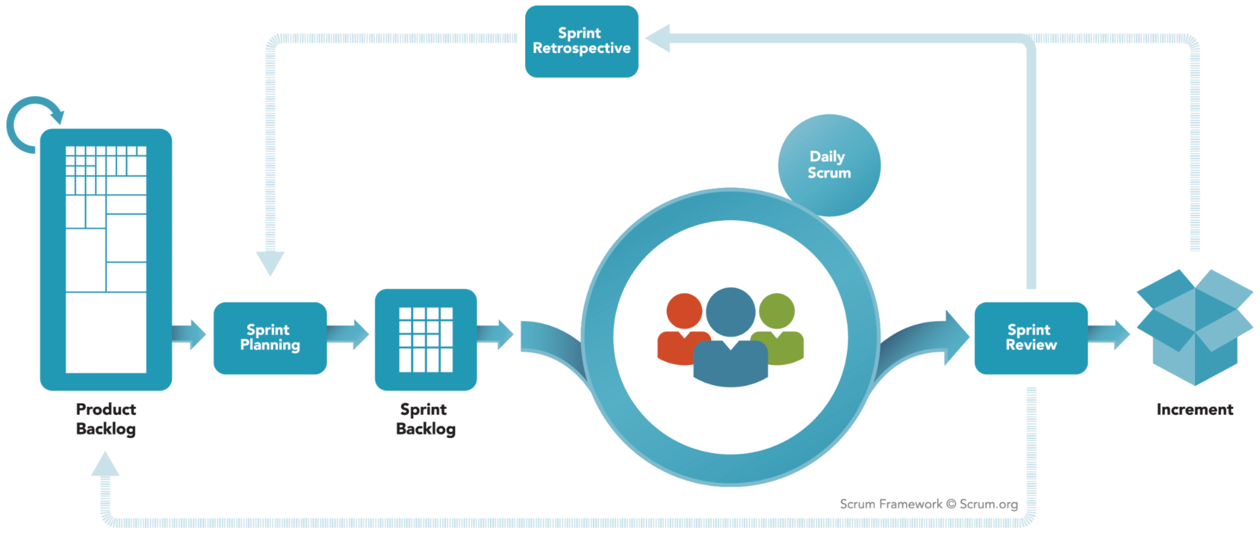
What is Scrum?

The scrum framework is heuristic; it's based on continuous learning and adjustment to fluctuating factors. It acknowledges that the team doesn't know everything at the start of a project and will evolve through experience. Scrum is structured to help teams naturally adapt to changing conditions and user requirements, with re-prioritization built into the process and short release cycles so your team can constantly learn and improve.



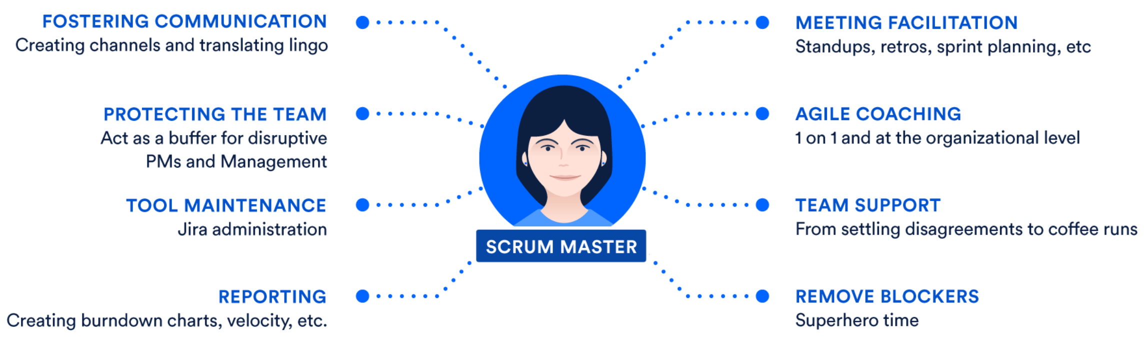
There are the three constants in a scrum team that we continue to revisit and invest in overtime.

- Product Backlog is the primary list of work that needs to get done maintained by the product owner or product manager. This is a dynamic list of features, requirements, enhancements, and fixes that acts as the input for the sprint backlog. It is, essentially, the team's "To Do" list.
- Sprint Backlog is the list of items, user stories, or bug fixes, selected by the development team for implementation in the current sprint cycle. Before each sprint, in the sprint planning meeting the team chooses which items it will work on for the sprint from the product backlog.
- Increment (or Sprint Goal) is the usable end-product from a sprint. At Atlassian, we usually demonstrate the "increment" during the end-of-sprint demo, where the team shows what was completed in the sprint. You may not hear the word "increment" out in the world, as it's often referred to as the team's definition of "Done", a milestone, the sprint goal, or even a full version or a shipped epic. It just depends on how your teams defines "Done" and how you define your sprint goals.



What is Scrum Master?

Scrum masters are the facilitators of scrum, the lightweight agile framework with a focus on time-boxed iterations called sprints. As facilitators, scrum masters act as coaches to the rest of the team. “Servant leaders” as the Scrum Guide puts it. Good scrum masters are committed to the scrum foundation and values, but remain flexible and open to opportunities for the team to improve their workflow.


FOSTERING COMMUNICATION
Creating channels and translating lingo
PROTECTING THE TEAM
Act as a buffer for disruptive PMs and Management
TOOL MAINTENANCE
Jira administration
REPORTING
Creating burndown charts, velocity, etc.
SCRUM MASTER
MEETING FACILITATION
Standups, retros, sprint planning, etc
AGILE COACHING
1 on 1 and at the organizational level
TEAM SUPPORT
From settling disagreements to coffee runs
REMOVE BLOCKERS
Superhero time

What is a sprint?

In scrum and other agile software development frameworks, a **sprint** is a repeatable fixed time-box during which a "Done" product of the highest possible value is created. Sprint lies at the core of the Scrum agile methodology and can be thought of as an event which wraps all other Scrum events like Daily Scrums, Scrum Review and Sprint Retrospective. Like all of scrum events, Sprint also has a maximum duration. Usually, a Sprint lasts for one month or less.[1]

Usually, daily meetings are held to discuss the progress of the project undertaken and any difficulty faced by any team member of the team while implementing the project. The outcome of the sprint is a deliverable, albeit with some increments. The scrum is used for projects like Web Technology or development of a product for the new market, i.e. the product with many requirements or fast-changing requirement.

A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies, and getting sprints right will help your agile team ship better software with fewer headaches.

Project Inception

Project Inception is a preliminary phase in most projects to explore the reason for starting a project, identify a solution for implementation, define benefits to be gained upon successful completion, estimate time required to do the project, and request for funds necessary for project execution. The purpose is to begin to define the overall project parameters and establish an appropriate management environment required to launch and manage the project.

What are product goals?

Product goals represent the crucial accomplishments needed to make your vision a reality. They highlight how the product is going to support the business and are often stepping stones to accelerating business growth.

Goals should be easy to understand, actionable, and achievable. They should also have a fixed time frame — typically lined up with fiscal planning cycles and spanning anywhere from three to 12 months.

As an example, consider a fictitious company called Fredwin Cycling that makes a cycling app. A few of their product goals include:

Goal: Top-rated social fitness cycling app within 12 months

Metric: #1 rated in iOS and Android marketplaces

Goal: Double revenue year over year

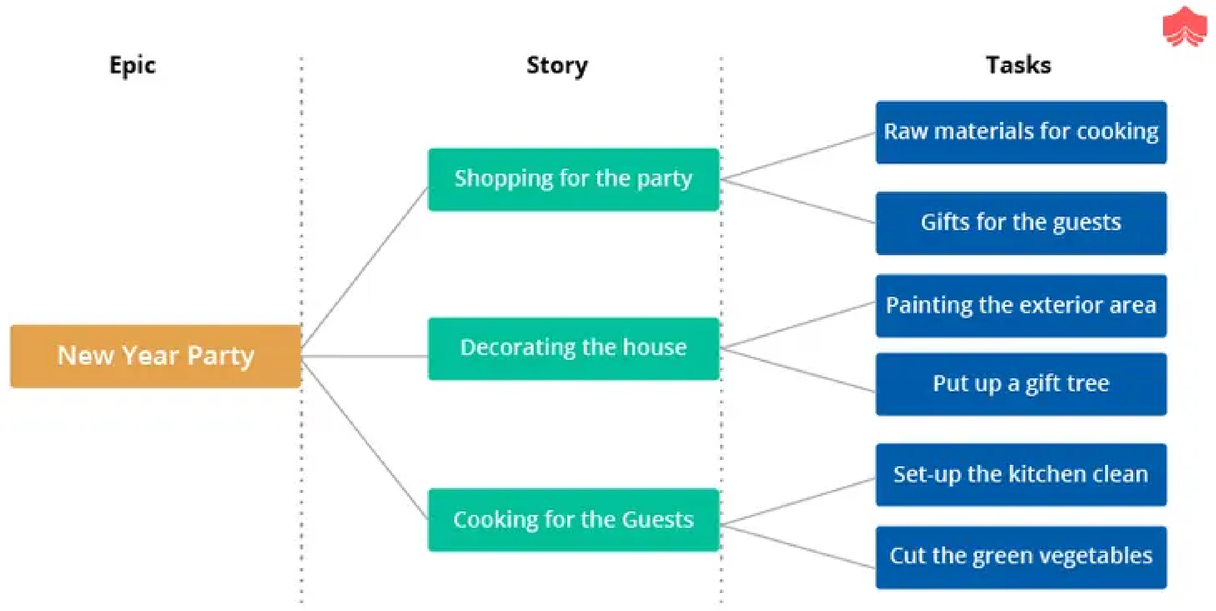
Metric: +$100M revenue

Goal: Largest partner ecosystem

Metric: +100 partners

The right goals will light a spark within the team — giving you a shared target to work towards and a sense of greater purpose. But simply setting goals is not enough. You also need to map all of the detailed work back to the goal it supports and track progress to let everyone see how their work reinforces success at a high level.

What is an Epic in Agile?
An epic is a large user story which is too big to fit into a sprint. This high-level story is usually split into smaller ones, each of which can be completed within a sprint. In that sense, an epic is a collection of user stories with a unified goal.
A requirement that is just too big to deliver in a single sprint. Epics need to be broken into smaller deliverables (stories).In simple terms, Scrum Epic in Agile Methodology is a big chunk of work which can be divided into smaller user stories. An Epic can be spread across sprints and even across agile teams. An Epic can be a high-level description of what the client wants, and accordingly, it has some value attached to it. As we mentioned, an Epic is a high-level requirement, hence its scope can change over the course of time.



You can think of an epic in two ways:

1.) The top-down view:

An epic is a body of work that a product team devises as they break down a strategic theme into smaller initiatives. A theme on your product roadmap might contain two or more epics.

2.) The bottom-up view:

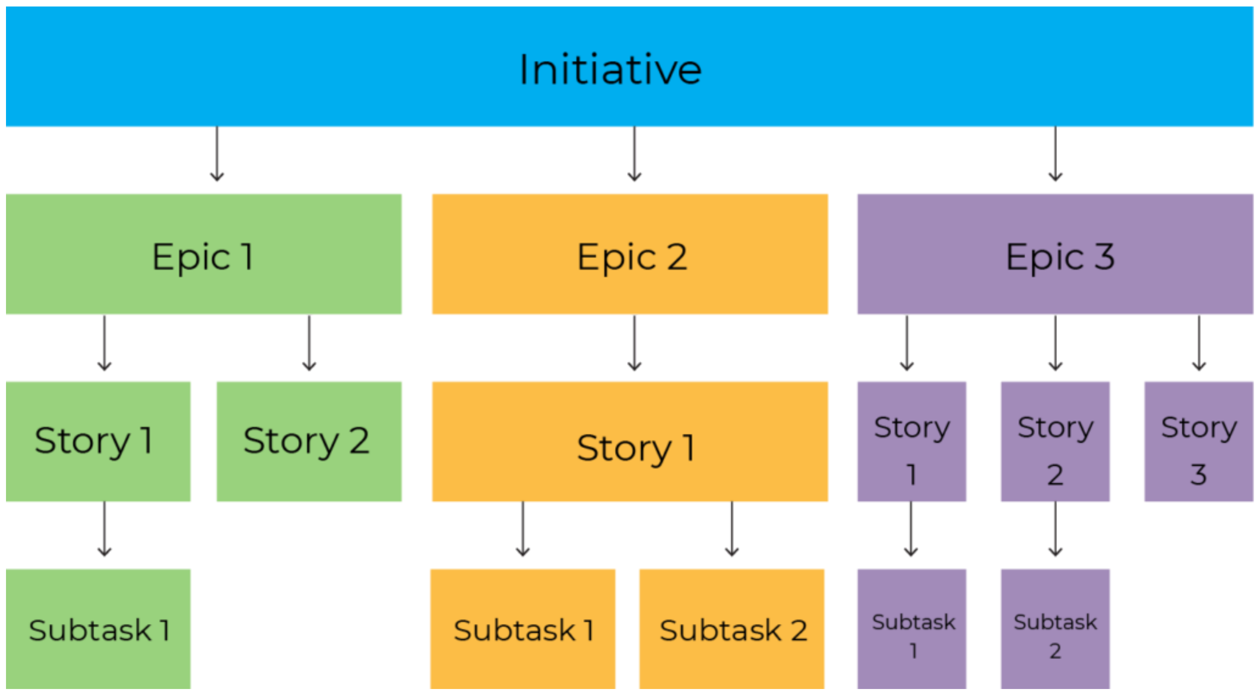
An epic is a body of work representing a group of user stories sharing a common strategic goal. Several related stories on the roadmap will often roll up to a single epic.

What is an Example of an Epic in Product Management?

Refer the below link
https://www.productplan.com/learn/how-to-write-an-epic/

Creating User Stories

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.



How to write user stories
Consider the following when writing user stories:

- Definition of "done" — The story is generally "done" when the user can complete the outlined task, but make sure to define what that is.
- Outline subtasks or tasks — Decide which specific steps need to be completed and who is responsible for each of them.
- User personas — For whom? If there are multiple end users, consider making multiple stories.
- Ordered Steps — Write a story for each step in a larger process.

- Listen to feedback — Talk to your users and capture the problem or need in their words. No need to guess at stories when you can source them from your customers.

- Time — Time is a touchy subject. Many development teams avoid discussions of time altogether, relying instead on their estimation frameworks.

Creating Test Plan
A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager."Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities."

How to write a Test Plan
You already know that making a **Test Plan** is the most important task of Test Management Process. Follow the seven steps below to create a test plan as per IEEE 829

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

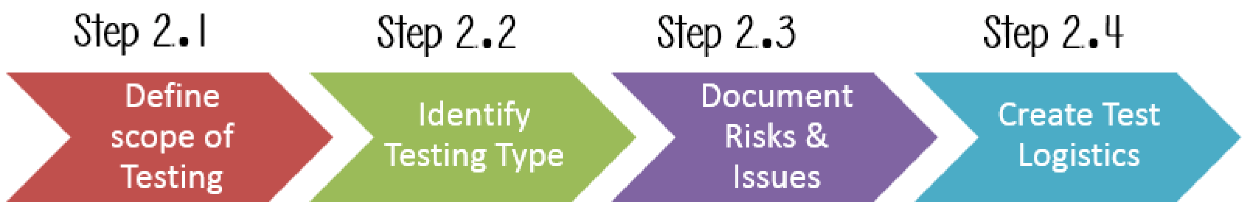
Step 1) Analyze the product
How can you test a product **without** any information about it? The answer is **Impossible.** You must learn a product **thoroughly** before testing it.
You should research clients and the end users to know their needs and expectations from the application

- Who will use the website?
- What is it used for?
- How will it work?
- What are software/ hardware the product uses?

Step 2) Develop Test Strategy
Test Strategy is a **critical step** in making a Test Plan in Software Testing. A Test Strategy document, is a high-level document, which is usually developed by Test Manager. This document defines:

- The project's **testing objectives** and the means to achieve them

- Determines testing **effort** and **costs**



Step 3) Define Test Objective
Test Objective is the overall goal and achievement of the test execution. The objective of the testing is finding as many software defects as possible; ensure that the software under test is **bug free** before release.
To define the test objectives, you should do 2 following steps

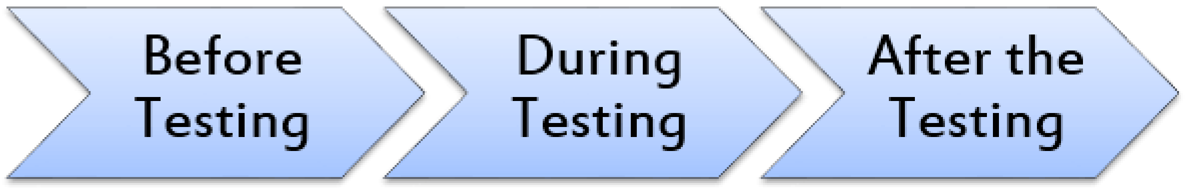
1. List all the software features (functionality, performance, GUI…) which may need to test.
2. Define the **target** or the **goal** of the test based on above features

Step 4) Define Test Criteria
Test Criteria is a standard or rule on which a test procedure or test judgment can be based.
There Are 2 types of test criteria as following
Suspension Criteria
Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be **suspended** until the criteria are **resolved**.
Test Plan Example: If your team members report that there are **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.
Exit Criteria
It specifies the criteria that denote a **successful** completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: **95%** of all critical test cases must pass.
Some methods of defining exit criteria are by specifying a targeted **run rate** and **pass rate**.
Step 5) Resource Planning
Resource plan is a **detailed summary** of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project
The resource planning is important factor of the test planning because helps in **determining** the **number** of resources (employee, equipment…) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.
Step 6) Plan Test Environment
A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of **real business** and **user** environment, as well as physical environments, such as server, front end running environments.
Step 7) Schedule & Estimation

In the article Test estimation, you already used some techniques to estimate the effort to complete the project. Now you should include that estimation as well as the schedule to the Test Planning
In the Test Estimation phase, suppose you break out the whole project into small tasks and add the estimation for each task as below

Task	Members	Estimate effort
Create the test specification	Test Designer	170 man-hour
Perform Test Execution	Tester, Test Administrator	80 man-hour
Test Report	Tester	10 man-hour
Test Delivery		20 man-hour
Total		**280 man-hour**

Step 8) Test Deliverables
Test Deliverables is a list of all the documents, tools and other components that has to be developed and maintained in support of the testing effort.
There are different test deliverables at every phase of the software development lifecycle.



What is Test Bed?

The test execution environment configured for testing. Test bed consists of specific hardware, software, Operating system, network configuration, the product under test, other system software and application software.

Test Bed Configuration:

It is the combination of hardware and software environment on which the tests will be executed. It includes hardware configuration, operating system settings, software configuration, test terminals and other support to perform the test.

Example:

A typical test bed for a web-based application is given below:
Web Server - IIS/Apache
Database - MS SQL
OS - Windows/ Linux
Browser - IE/FireFox
Java version : version 6

What is Test Data in Software Testing?
Test Data in Software Testing is the input given to a software program during test execution. It represents data that affects or affected by software execution while testing. Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs.

What is a Test case?

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Typical Test Case Parameters:

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Example:

Let us say that we need to check an input field that can accept maximum of 10 characters.

While developing the test cases for the above scenario, the test cases are documented the following way. In the below example, the first case is a pass scenario while the second case is a FAIL.

Scenario	**Test Step**	**Expected Result**	**Actual Outcome**
Verify that the input field that can accept maximum of 10 characters	Login to application and key in 10 characters	Application should be able to accept all 10 characters.	Application accepts all 10 characters.
Verify that the input field that can accept maximum of 11 characters	Login to application and key in 11 characters	Application should NOT accept all 11 characters.	Application accepts all 10 characters.

If the expected result doesn't match with the actual result, then we log a defect. The defect goes through the defect life cycle and the testers address the same after fix.

Scope of Product
Product scope refers to the number of different items your company offers for sale. Your business goals usually determine the scope of products you carry. You may run a successful business based on a single product strategy or offer a much deeper line of products to serve a wider range of customers. Your product scope determines your future marketing strategies, profit goals and territory saturation.

Iteration Planning / Sprint Planning
Iteration planning is the process by which planning is done on how many tasks can be lined up to make the feature in a single sprint. Sometimes a single feature consists of multiple sprint iterations. The owner and the master based on the resources decide how many tasks can be committed in a single iteration.

Iteration planning is helpful to make the sprint and the product backlog more effective and the development activities of the tasks more efficient. Planning is the key to success. At every stage one must meticulously plan to deliver the right outcome. The repetitive process became successful in development due to the right planning at the right time. The entire team sits together to plan and commit about how much they can complete.

When is iteration Planning completed?
Logically speaking the iteration planning is completed only when the last sprint backlog is committed with the final sets of tasks. However, each set of sprint backlogs consists of iteration planning and is completed on the completion of that particular sprint.

1) Inputs to the iteration planning – These are nothing but the tasks that can be accomplished by the team in the single iteration that is a single sprint. This is planned using the user stories, during the sequencing and scheduling of the user stories and taking in to account the previous projects or the historical averages, from lessons learned from previous projects, taking into account any similar projects completed previously and by studying any other similar team in the other projects. One other important parameter is the system demo that was done before the start of the project.

2) Planning the iteration – This is done when the product owner establishes the program increments and then breaks down the requirements into user stories for the resources to work on.

3) Capacity estimation – This is nothing but calculating the time required by the resources such as velocity and story points calculations. For example – If a sprint is said to be for one week and then there are 4 developers and 3 testers and 1 Product owner then the velocity is calculated as 5 * 8 that is 40 story points estimation per iteration and so on.

4) Story Analysis and estimation – This is done by the product owner when each user story and epics are made in the product backlog depending on the complexity, how long it will take to complete, how many resources are needed, size, difficulty, acceptance criteria, technical challenges, and uncertainty. This is also called the Behavior-Driven Development (BDD) of the tasks in Scrum.

5) User stories – This is nothing but making the individual tasks from the user's stories for the resources to act upon and complete the tasks in a sprint backlog and then planning many more iterations if required.

6) Goals of the iteration – Making the iteration goals to develop a feature of the product is the next step. The iteration goals will accomplish certain customer requirements in an iteration.

7) Finally committing to the iteration goals – Once the goals of the single iteration are established and worked upon then committing to the iteration goals are the final steps in the iteration planning stages.

Schedule of the Project
Project scheduling involves creating a document, these days usually a digital document, that details the project timeline and the organizational resources required to complete each task.
The project schedule must be accessible to every team member. Its purpose is to communicate critical information to the team, so it must be comprehensive and easy to understand.
A project schedule indicates what needs to be done, which resources must be utilized, and when the project is due. It's a timetable that outlines start and end dates and milestones that must be met for the project to be completed on time. The project schedule is often used in conjunction with a work breakdown structure (WBS) to distribute work among team members. The project schedule should be updated regularly to gain a better understanding of the project's status.

Cost Estimation
Cost Estimation is a statement that gives the value of the cost incurred in the manufacturing of finished goods. Cost estimation helps in fixing the selling price of the final product after

charging appropriate overheads and allowing a certain margin for profits. Cost estimation in project management is the process of forecasting the financial and other resources needed to complete a project within a defined scope. Cost estimation accounts for each element required for the project—from materials to labor—and calculates a total amount that determines a project’s budget. An initial cost estimate can determine whether an organization greenlights a project, and if the project moves forward, the estimate can be a factor in defining the project’s scope. If the cost estimation comes in too high, an organization may decide to pare down the project to fit what they can afford

Elements of cost estimation in project management

There are two key types of costs addressed by the cost estimation process:

1. **Direct costs:** Costs associated with a single area, such as a department or the project itself. Examples of direct costs include fixed labor, materials, and equipment.

2. **Indirect costs:** Costs incurred by the organization at large, such as utilities and quality control.

Within these two categories, here are some typical elements that a cost estimation will take into account:

- **Labor:** The cost of team members working on the project, both in terms of wages and time
- **Materials and equipment:** The cost of resources required for the project, from physical tools to software to legal permits
- **Facilities:** The cost of using any working spaces not owned by the organization.
- **Vendors:** The cost of hiring third-party vendors or contractors.

- **Risk:** The cost of any contingency plans implemented to reduce risk.

WHAT IS A BURNDOWN CHART?

A burndown chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining. Burndown charts are used to predict your team's likelihood of completing their work in the time available. They're also great for keeping the team aware of any scope creep that occurs.

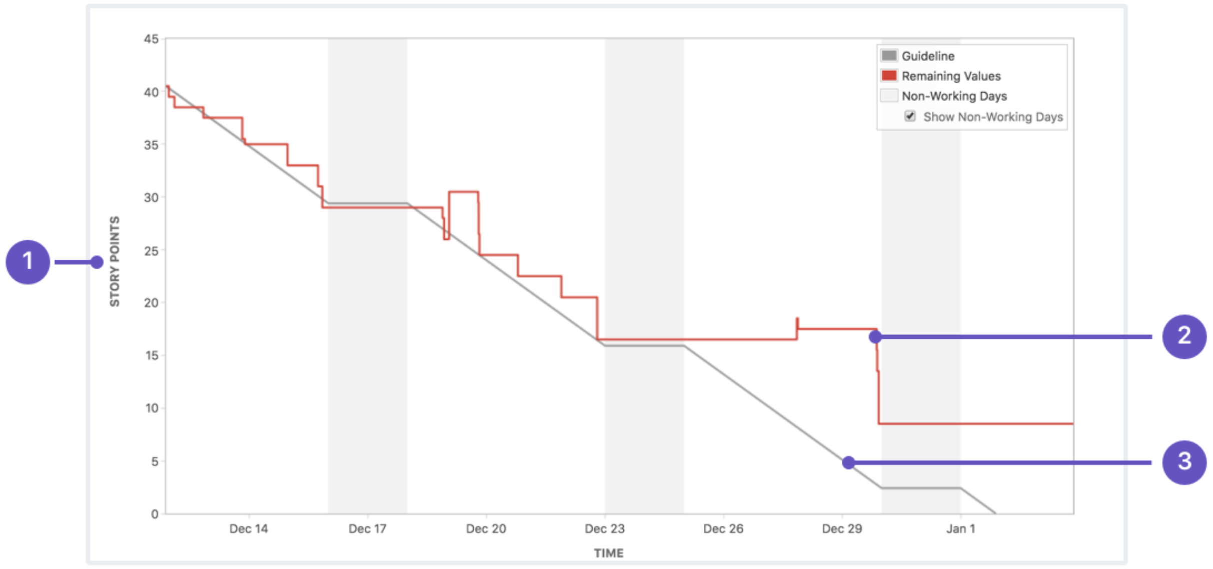
Burndown charts are useful because they provide insight into how the team works. For example:

- If you notice that the team consistently finishes work early, this might be a sign that they aren't committing to enough work during sprint planning.
- If they consistently miss their forecast, this might be a sign that they've committed to too much work.

- If the burndown chart shows a sharp drop during the sprint, this might be a sign that work has not been estimated accurately, or broken down properly.

This report shows the amount of work to be done in a sprint. It can be used to track the total work remaining in the sprint, and to project the likelihood of achieving the sprint goal. By tracking the remaining work throughout the sprint, a team can manage its progress, and respond to trends accordingly. For example, if the burndown chart shows that the team may not reach the sprint goal, then they can take the necessary actions to stay on track.

Understanding the sprint burndown chart



1. Estimation statistic: The vertical axis represents the estimation statistic that you've selected.
2. Remaining values: The red line represents the total amount of work left in the sprint, according to your team's estimates.
3. Guideline: The grey line shows an approximation of where your team should be, assuming linear progress.

Risk Management

Risk management is the process of identifying, assessing and controlling financial, legal, strategic and security risks to an organization’s capital and earnings. These threats, or risks, could stem from a wide variety of sources, including financial uncertainty, legal liabilities, strategic management errors, accidents and natural disasters.

If an unforeseen event catches your organization unaware, the impact could be minor, such as a small impact on your overhead costs. In a worst-case scenario, though, it could be catastrophic and have serious ramifications, such as a significant financial burden or even the closure of your business.

Three important steps of the risk management process are risk identification, risk analysis and assessment, and risk mitigation and monitoring.

Identifying risks
Risk identification is the process of identifying and assessing threats to an organization, its operations and its workforce. For example, risk identification may include assessing IT security threats such as malware and ransomware, accidents, natural disasters and other potentially harmful events that could disrupt business operations.
Risk analysis and assessment
Risk analysis involves establishing the probability that a risk event might occur and the potential outcome of each event. Risk evaluation compares the magnitude of each risk and ranks them according to prominence and consequence.
Risk mitigation and monitoring
Risk mitigation refers to the process of planning and developing methods and options to reduce threats to project objectives. A project team might implement risk mitigation strategies to identify, monitor and evaluate risks and consequences inherent to completing a specific project, such as new product creation. Risk mitigation also includes the actions put into place to deal with issues and effects of those issues regarding a project.
Risk management is a nonstop process that adapts and changes over time. Repeating and continually monitoring the processes can help assure maximum coverage of known and unknown risks.

What are the most common responses to risk?
Risk avoidance
Avoidance is a method for mitigating risk by not participating in activities that may negatively affect the organization. Not making an investment or starting a product line are examples of such activities as they avoid the risk of loss.
Risk reduction
This method of risk management attempts to minimize the loss, rather than completely eliminate it. While accepting the risk, it stays focused on keeping the loss contained and preventing it from spreading. An example of this in health insurance is preventative care.
Risk sharing
When risks are shared, the possibility of loss is transferred from the individual to the group. A corporation is a good example of risk sharing — a number of investors pool their capital and each only bears a portion of the risk that the enterprise may fail.
Transferring risk
Contractually transferring a risk to a third-party, such as, insurance to cover possible property damage or injury shifts the risks associated with the property from the owner to the insurance company.
Risk acceptance and retention
After all risk sharing, risk transfer and risk reduction measures have been implemented, some risk will remain since it is virtually impossible to eliminate all risk (except through risk avoidance). This is called residual risk.

Software Design Principles

Software design principles are concerned with providing means to handle the complexity of the design process effectively. Effectively managing the complexity will not only reduce the effort needed for design but can also reduce the scope of introducing errors during design.

Availability

Availability is defined as the probability that the system is operating properly when it is requested for use. In other words, availability is the probability that a system is not failed or undergoing a repair action when it needs to be used. At first glance, it might seem that if a system has a high availability then it should also have a high reliability. However, this is not necessarily the case. This article will explore the relationship between availability and reliability and will also present some of the specified classifications of availability. In fact, availability builds upon the concept of reliability by adding the notion of recovery—that is, when the system breaks, it repairs itself.

Performance

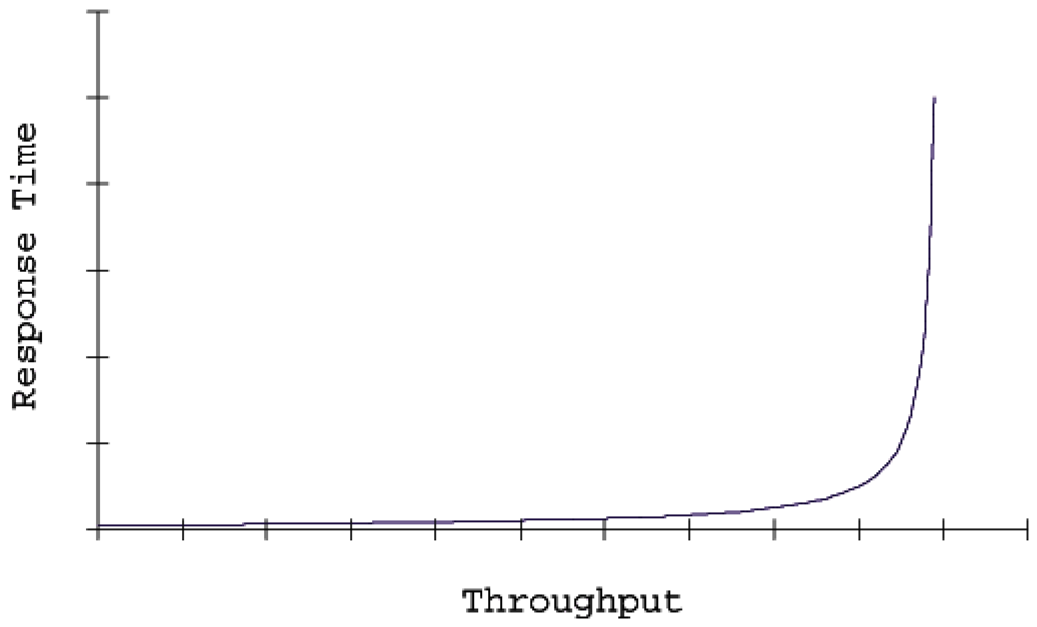
Performance is an indicator of how well a software system or component meets its requirements for timeliness. Timeliness is measured in terms of response time or throughput. The *response time* is the time required to respond to a request. It may be the time required for a single transaction, or the end-to-end time for a user task. For example, we may require that an online system provide a result within one-half second after the user presses the "enter" key.

Consistency

Consistency in design means we produce all the elements with similarity. They should look and work as the parts of one bigger organism. This way, we give the user a much easier and more pleasant experience with our final product. Consistency is crucial to create intuitive mobile apps and websites.
It really matters, when it comes to learning things – for example how to use new software. With consistent design finding out how it works and where to find functionalities we are interested in is easy-peasy. **Consistency just provides context that is understandable for most of us, so we can transfer our knowledge from one product we use to another.**

Scalability

Scalability is the ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases.The graph in Figure 1-1 illustrates how increasing use of a system affects its response time.



As you can see from the curve, as long as you are below a certain threshold, increasing the load does not have a great effect on response time. In this region, the response time increases linearly with the load. At some point, however, a small increase in load begins to have a great effect on response time. In this region (at the right of the curve), the response time increases exponentially with the load.

Manageability

How efficiently and easily a software system can be monitored and maintained to keep the system performing, secure, and running smoothly. In general, manageability is the measure of and set of features that support the ease, speed, and competence with which a system can be discovered, configured, modified, deployed, controlled, and supervised.

Cost

For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take. These estimates are needed before development is initiated.

Uses of Cost Estimation

1. During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.

2. In monitoring the project's progress, one needs to access whether the project is progressing according to the procedure and takes corrective action, if necessary.

Software Architecture :

Software architecture is the blueprint of building software. It shows the overall structure of the software, the collection of components in it, and how they interact with one another while hiding the implementation.

This helps the software development team to clearly communicate how the software is going to be built as per the requirements of customers.

Different Software Architecture Patterns :

1. Layered Pattern
2. Monolithic
3. Service oriented
4. Microservices Pattern

1. Layered Pattern :

As the name suggests, components(code) in this pattern are separated into layers of subtasks and they are arranged one above another.

Each layer has unique tasks to do and all the layers are independent of one another. Since each layer is independent, one can modify the code inside a layer without affecting others.

It is the most commonly used pattern for designing the majority of software. This layer is also known as 'N-tier architecture'. Basically, this pattern has 4 layers.

1. Presentation layer (The user interface layer where we see and enter data into an application.)
2. Business layer (this layer is responsible for executing business logic as per the request.)
3. Application layer (this layer acts as a medium for communication between the 'presentation layer' and 'data layer'.

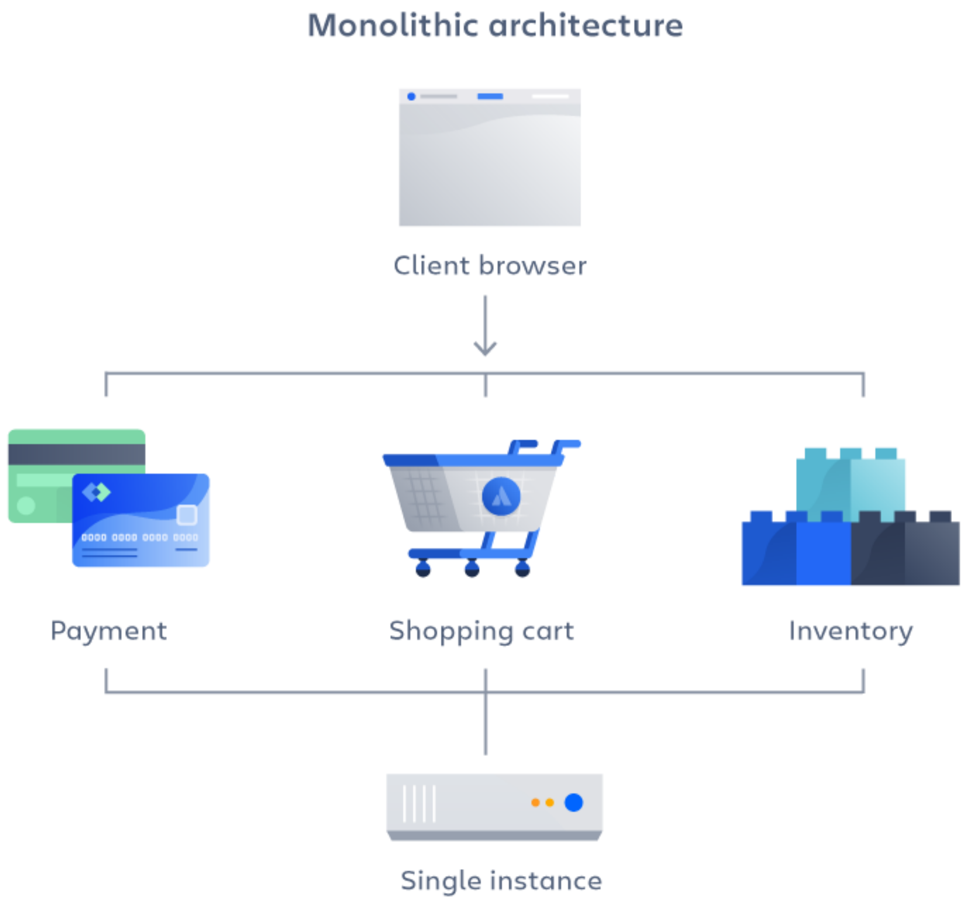
4. Data layer (this layer has a database for managing data.)

Ideal for:

E-commerce web applications development like Amazon.

2. Monolithic

A monolithic architecture is a traditional model of a software program, which is built as a unified unit that is self-contained and independent from other applications. The word “monolith” is often attributed to something large and glacial, which isn’t far from the truth of a monolith architecture for software design. A monolithic architecture is a singular, large computing network with one code base that couples all of the business concerns together. To make a change to this sort of application requires updating the entire stack by accessing the code base and building and deploying an updated version of the service-side interface. This makes updates restrictive and time-consuming.



The advantages of a monolithic architecture include:

Easy deployment – One executable file or directory makes deployment easier.

Development – When an application is built with one code base, it is easier to develop.

Performance – In a centralized code base and repository, one API can often perform the same function that numerous APIs perform with microservices.

Simplified testing – Since a monolithic application is a single, centralized unit, end-to-end testing can be performed faster than with a distributed application.

Easy debugging – With all code located in one place, it's easier to follow a request and find an issue.

The disadvantages of a monolith include:

Slower development speed – A large, monolithic application makes development more complex and slower.

Scalability – You can't scale individual components.

Reliability – If there's an error in any module, it could affect the entire application's availability.

3. Service Oriented Architecture

Service-oriented architecture (SOA) is a method of software development that uses software components called services to create business applications. Each service provides a business capability, and services can also communicate with each other across platforms and languages. Developers use SOA to reuse services in different systems or combine several independent services to perform complex tasks.

For example, multiple business processes in an organization require the user authentication functionality. Instead of rewriting the authentication code for all business processes, you can create a single authentication service and reuse it for all applications. Similarly, almost all systems across a healthcare organization, such as patient management systems and electronic health record (EHR) systems, need to register patients. These systems can call a single, common service to perform the patient registration task.

Some major benefits of SOA include the following:

Faster time to market

Developers reuse services across different business processes to save time and costs. They can assemble applications much faster with SOA than by writing code and performing integrations from scratch.

Efficient maintenance

It’s easier to create, update, and debug small services than large code blocks in monolithic applications. Modifying any service in SOA does not impact the overall functionality of the business process.

Greater adaptability

SOA is more adaptable to advances in technology. You can modernize your applications efficiently and cost effectively. For example, healthcare organizations can use the functionality of older electronic health record systems in newer cloud-based applications.

4. Microservices Pattern :

The collection of small services that are combined to form the actual application is the concept of microservices pattern. Instead of building a bigger application, small programs are built for every service (function) of an application independently. And those small programs are bundled together to be a full-fledged application.

So adding new features and modifying existing microservices without affecting other microservices are no longer a challenge when an application is built in a microservices pattern.

Modules in the application of microservices patterns are loosely coupled. So they are easily understandable, modifiable and scalable.

Example Netflix is one of the most popular examples of software built-in microservices architecture. This pattern is most suitable for websites and web apps having small components.

Design methods for security

Security by design is an approach to software and hardware development that seeks to make systems as free of vulnerabilities and impervious to attack as possible through such measures as continuous testing, authentication safeguards and adherence to best programming practices. Security by design is rapidly becoming crucial in the rapidly developing Internet of Things (IoT) environment, in which almost any conceivable device, object or entity can be given a unique identifier (UID) and networked to make them addressable over the Internet.

Application Security

Application security describes security measures at the application level that aim to prevent data or code within the app from being stolen or hijacked. It encompasses the security considerations that happen during application development and design, but it also involves systems and approaches to protect apps after they get deployed.

Application security may include hardware, software, and procedures that identify or minimize security vulnerabilities. A router that prevents anyone from viewing a computer's IP address from the Internet is a form of hardware application security. But security measures at the application level are also typically built into the software, such as an application firewall that strictly defines what activities are allowed and prohibited. Procedures can entail things like an application security routine that includes protocols such as regular testing.

Authentication is the process of verifying the identity of a user or information. User authentication is the process of verifying the identity of a user when that user logs in to a computer system. Authentication is used by a server when the server needs to know exactly who is accessing their information or site. Usually, authentication by a server entails the use of a user name and password. Other ways to authenticate can be through cards, retina scans, voice recognition, and fingerprints.

The list below reviews some common authentication methods used to secure modern systems.

Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features. Authorization is normally preceded by authentication for user identity verification. System administrators (SA) are typically assigned permission levels covering all system and user resources.

Authentication and Authorisation Methods

1. Token- Based Authentication

A Token is a computer-generated code that acts as a digitally encoded signature of a user. They are used to authenticate the identity of a user to access any website or application network. Token-based authentication is a two-step authentication strategy to enhance the security mechanism for users to access a network. The users once register their credentials, receive a unique encrypted token that is valid for a specified session time. During this session, users can directly access the website or application without login requirements.

2. Multi-factor authentication

Multi-Factor Authentication (MFA) is an authentication method that requires two or more independent ways to identify a user. Examples include codes generated from the user's smartphone, Captcha tests, fingerprints, voice biometrics or facial recognition.

MFA authentication methods and technologies increase the confidence of users by adding multiple layers of security.

3. Certificate-based authentication

Certificate-based authentication technologies identify users, machines or devices by using digital certificates. A digital certificate is an electronic document based on the idea of a driver's license or a passport.

The certificate contains the digital identity of a user including a public key, and the digital signature of a certification authority. Digital certificates prove the ownership of a public key and issued only by a certification authority.

Users provide their digital certificates when they sign in to a server. The server verifies the credibility of the digital signature and the certificate authority. The server then uses cryptography to confirm that the user has a correct private key associated with the certificate.

4. Biometric authentication

Biometrics authentication is a security process that relies on the unique biological characteristics of an individual. Biometric authentication technologies are used by consumers, governments and private corporations including airports, military bases, and national borders. The technology is increasingly adopted due to the ability to achieve a high level of security without creating friction for the user. Common biometric authentication methods include:

- **Facial recognition**—matches the different face characteristics of an individual trying to gain access to an approved face stored in a database.
- **Fingerprint scanners**—match the unique patterns on an individual's fingerprints. Some new versions of fingerprint scanners can even assess the vascular patterns in people's fingers.
- **Speaker Recognition** —also known as voice biometrics, examines a speaker's speech patterns for the formation of specific shapes and sound qualities.
- **Eye scanners**—include technologies like iris recognition and retina scanners. Iris scanners project a bright light towards the eye and search for unique patterns in the colored ring around the pupil of the eye.