Introduction SDLC, Agile

SDLC is the series of steps that we go through when creating new products.
It encompasses,
1. **Methodologies** *(broader categories of development concepts and practices) and*
2. **Frameworks** *(a more detailed implementation of a methodology's ideas).*
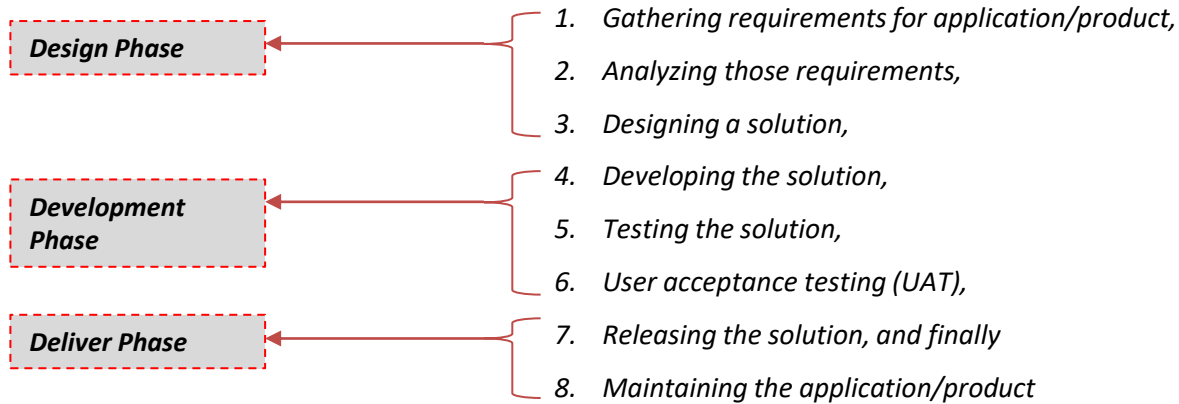
Methodologies-
1. **Waterfall methodology**
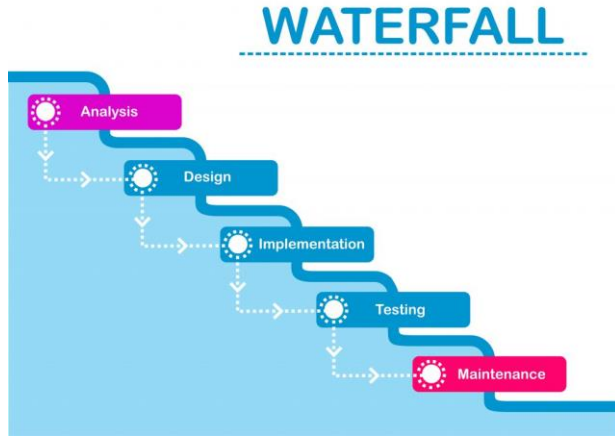2. **Agile methodology**

Agile Methodology Frameworks-
1. **Scrum**
2. **Kanban**
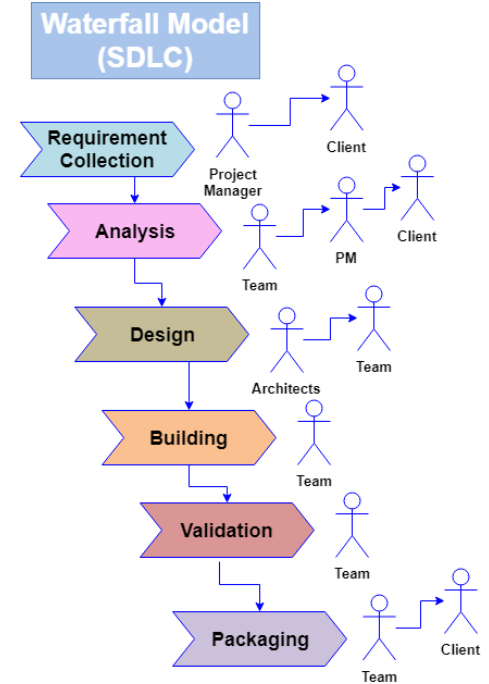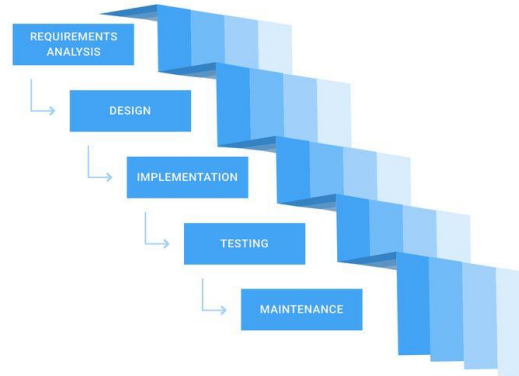3. **Scrumban**
4. **eXtreme Programming**

Basic Steps

Although the order/repetition of the steps in the two methodologies is different, they both contain the same overarching steps that are characteristic of SDLC as a whole

Design Phase

1. *Gathering requirements for application/product,*
2. *Analyzing those requirements,*
3. *Designing a solution,*

Development Phase

4. *Developing the solution,*
5. *Testing the solution,*
6. *User acceptance testing (UAT),*

Deliver Phase

7. *Releasing the solution, and finally*
8. *Maintaining the application/product*

Waterfall methodology

Waterfall practitioners follow each of the eight SDLC steps in order, never revisiting a previous stage; just like a natural waterfall, the methodology only flows in one direction.

Once development has begun, Waterfall practitioners accept no new requirements from the client.

Waterfall Benifits
- *There is no room for confusion in Waterfall because there is a **clear barrier between each step**, making it easy for team members to keep track of where they are in the project. This clear barrier also allows for a clean transfer of knowledge between steps. For example, when moving from development to testing, team members will already know every aspect of the project that will ever need to be tested; because there is no revisiting past steps, they do not have to worry about any future functionality in need of scrutiny. This clarity also applies to the entire Waterfall lifecycle - programmers have an understanding of what the finished product will look like and do as soon as the design phase has been completed because all requirements are gathered before design (or development, for that matter) begins.*
- *Waterfall does not require any specific procedural knowledge - followers simply intuitively move from one step and phase to the next.*

Waterfall Drawbacks
- ***Excludes the client** in every step between requirement gathering and deployment - your client better have given you a complete set of requirements at the beginning because once development starts, you're not accepting any new feature requests.*
- *The **one-direction flow** makes changes to previous stages impossible.*
- *Waterfall practitioners **do not test until all development has been completed**, so a bug or problem that may have been developed months prior will not be noticed until the entire project has entered the testing phase.*

What is Agile?

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.
It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two weeks cycles called iterations.

Scrum is an innovative approach to getting work done in efficient way. It is iterative & incremental agile software development method. These iterations are time boxed with various iterations & each iteration is called Sprint. The Sprint is basically 2-4 weeks long & each sprint requires sprint planning estimation. According to latest surveys Scrum is the most popular agile project management methodology in software development.

Where can we use Scrum?
Scrum is ideally used where highly emergent
*or rapidly changing requirements.*

SCRUM PROCESS

The Scrum Team
The Scrum Team consists of a Product Owner, the
Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional.
Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team.
Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team.
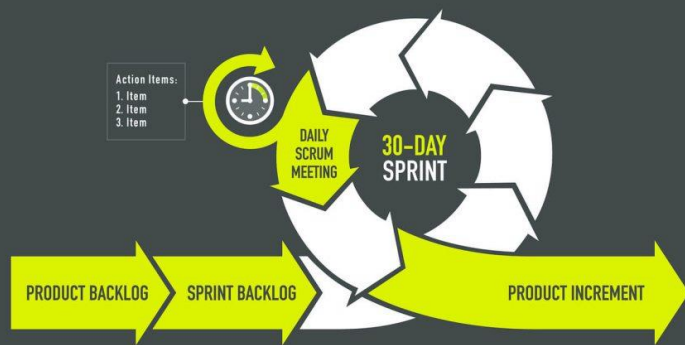
Before Jumping to Agile Framework

Theme: *A theme is a wide area of focus that helps an Agile team to keep track of their organizational goals — think of it as a label that can be used to group similar activities. A theme helps to define the common characteristics between different areas and unite them under one heading. A theme provides a convenient way to indicate that a set of related epics have something in common, such as being in the same functional area. By assigning a financial value to Themes, managers can ensure the highest value is being delivered and that the project/program is aligned with its objectives and the strategic direction of the organization.*

Epic*: A group of related features that is broken down into multiple user stories. An epic is a substantial collection of smaller stories that combine to make one large story. An epic cannot be completed in a single Agile iteration (or sprint). The key element to an epic is that it takes a lot of time.*

User Story*: An individual feature of/requirement for a project in Agile development. A story, also referred to as a user story, is a short-form request that can be delivered in one sprint. It is written in simple language from the perspective of the user.*

Tasks: *A task is a subsection of a story. It helps to break the story down and outline how it will be completed. Tasks tend to be more technical as they are used by members of the development team (e.g., a quality assurance tester) rather than a client/end user. Tasks are decomposed parts of a story that get into HOW the story will be completed. Tasks can be hour estimated if desired. Tasks are usually defined by the people doing the work (developers, QA, etc), whereas stories and epics are generally created by the customer or the product owner on behalf of the customer. Thus, the tasks no longer need to be understandable by business users and so can be highly technical. The process of breaking a story down into tasks also helps the development team better understand what needs to be done.*
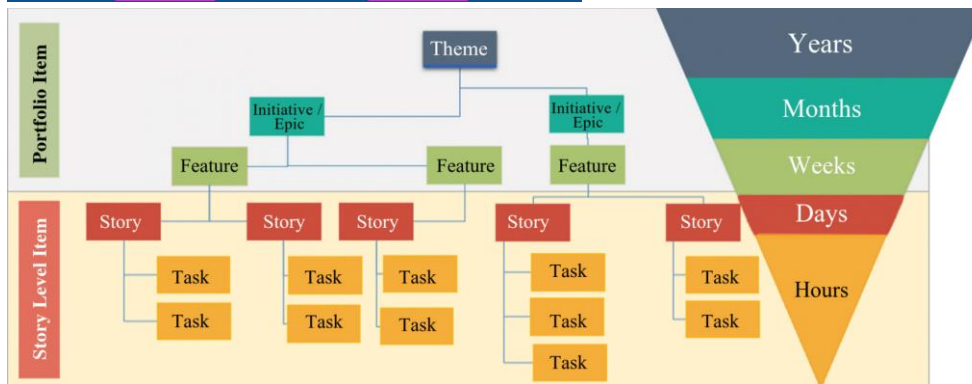
Before Jumping to Agile Framework

Initiative: *An initiative is a group of epics. It can incorporate epics from lots of different teams, but they will all have a common objective. An initiative will naturally take more time than an epic.*

Feature: *A feature refers to a certain functionality or service that satisfies a stakeholder's need. Each feature must outline the criteria involved and offer a specific business value.*

Initiative vs feature
Initiatives and features can both draw from multiple areas to create compilations. An initiative can include epics from different teams, and a feature can involve several different stories.
The key differentiator between the two is the amount of time required. Initiatives involve long-term planning and can take up to a year to complete. Features, however, will typically take place within a single quarter.

Before Jumping to Agile Framework

Sprint: A brief period of development (almost always less than four weeks, sometimes as short as one week) generally culminating in a release of related features.
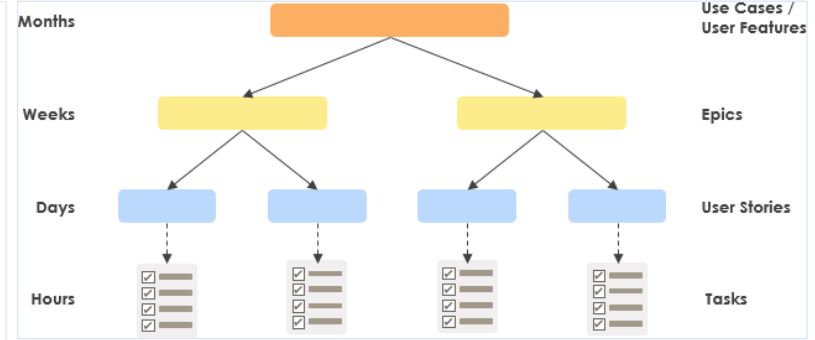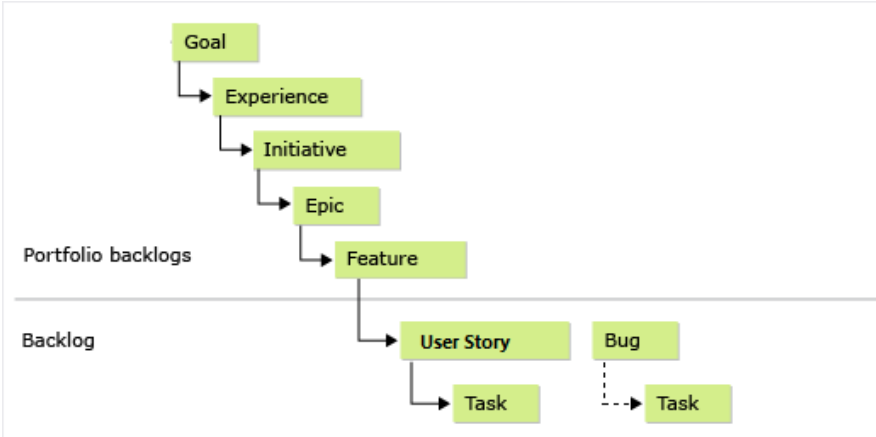
Story Point: A level of difficulty assigned to a user story through the use of a sequence of numbers that increases with increasing difficulty. Story points are used to measure the complexity of a story. The overall goal of a story is to provide value to its user within a set timeframe.

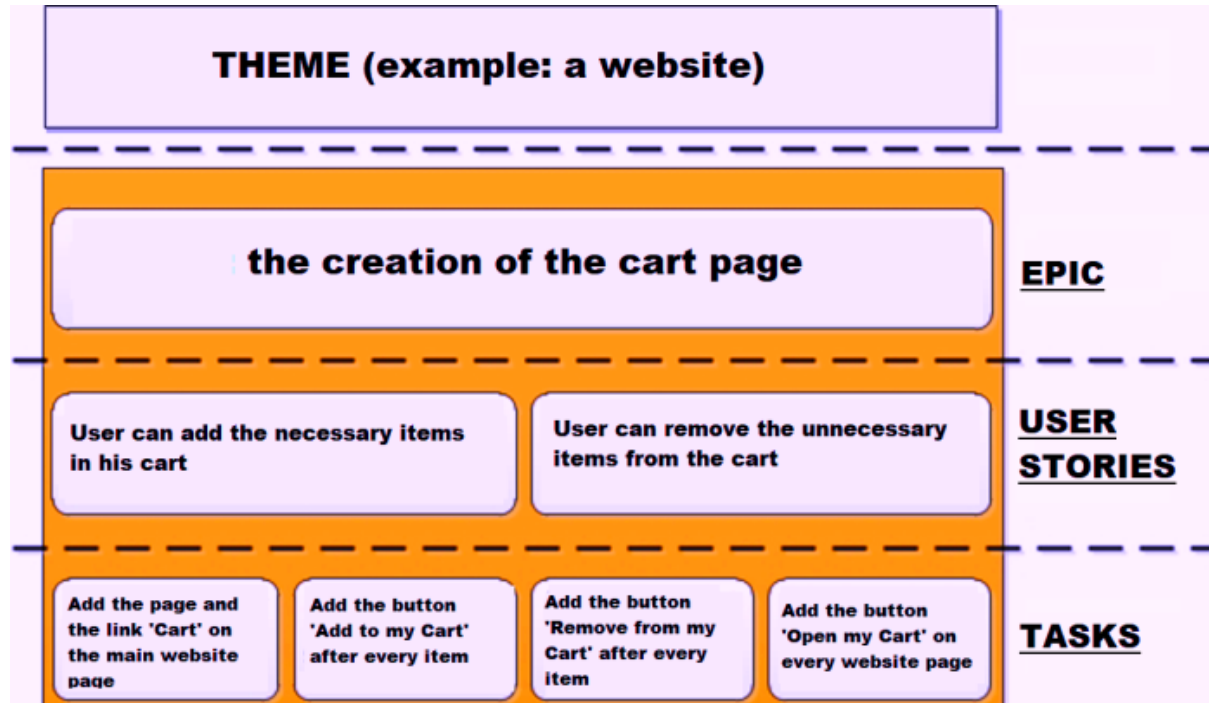*Velocity: The sum of story points of all user stories completed during a sprint. Velocity allows Agile teams to more accurately predict how many user stories can be completed in future sprints.*

Burndown Chart:
A burndown chart or burn down chart is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal.

Theme -> Epic -> Feature -> User Story -> Task

THEME (example: a website)

the creation of the cart page — EPIC

User can add the necessary items in his cart

User can remove the unnecessary items from the cart

USER STORIES

Add the page and the link 'Cart' on the main website page

Add the button 'Add to my Cart' after every item

Add the button 'Remove from my Cart' after every item

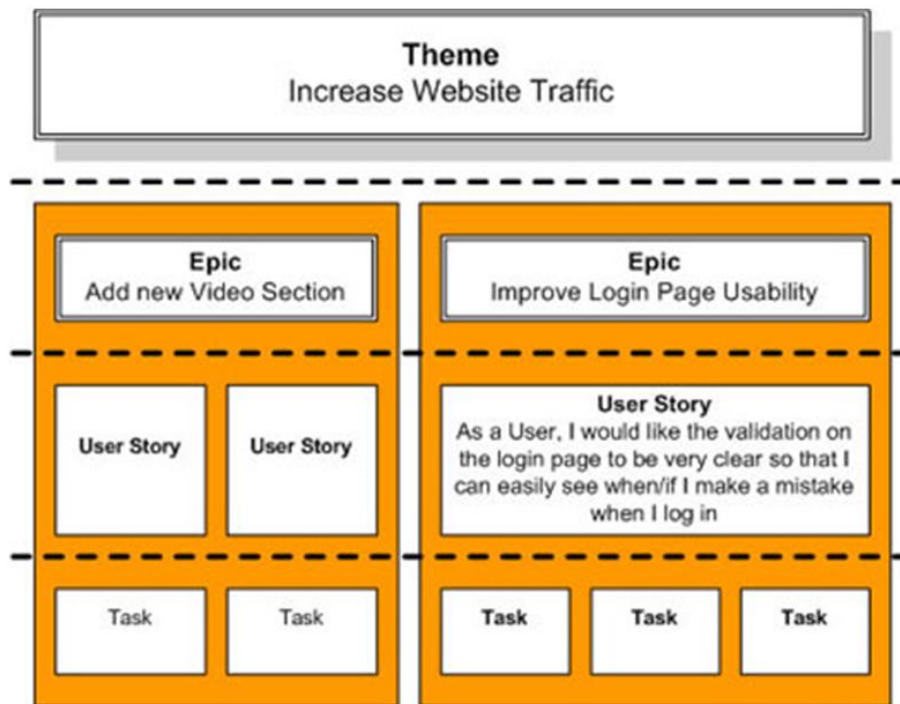Add the button 'Open my Cart' on every website page

TASKS

Theme vs Epic vs User Stories

Epic vs Story
A story is a single requirement, while an epic is a group of multiple stories. Picture a project with a lot of folders. The folders are the individual stories that are related in some way. They all combine to form one large project, which is the epic.

Another key difference between an epic and a story is the length of time each one takes. A story takes place within a sprint, which is usually about one or two weeks long. On the other hand, an epic will likely take between one and three months to complete.
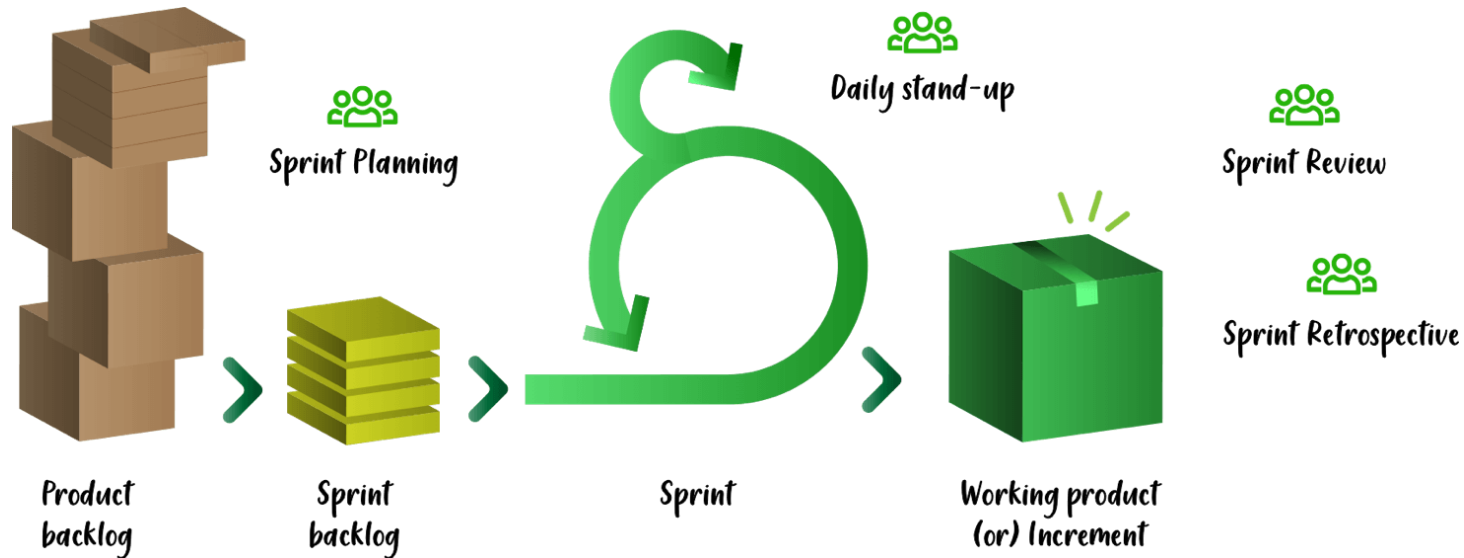
Theme
Increase Website Traffic

Epic
Add new Video Section

Epic
Improve Login Page Usability

User Story

User Story

User Story
As a User, I would like the validation on the login page to be very clear so that I can easily see when/if I make a mistake when I log in

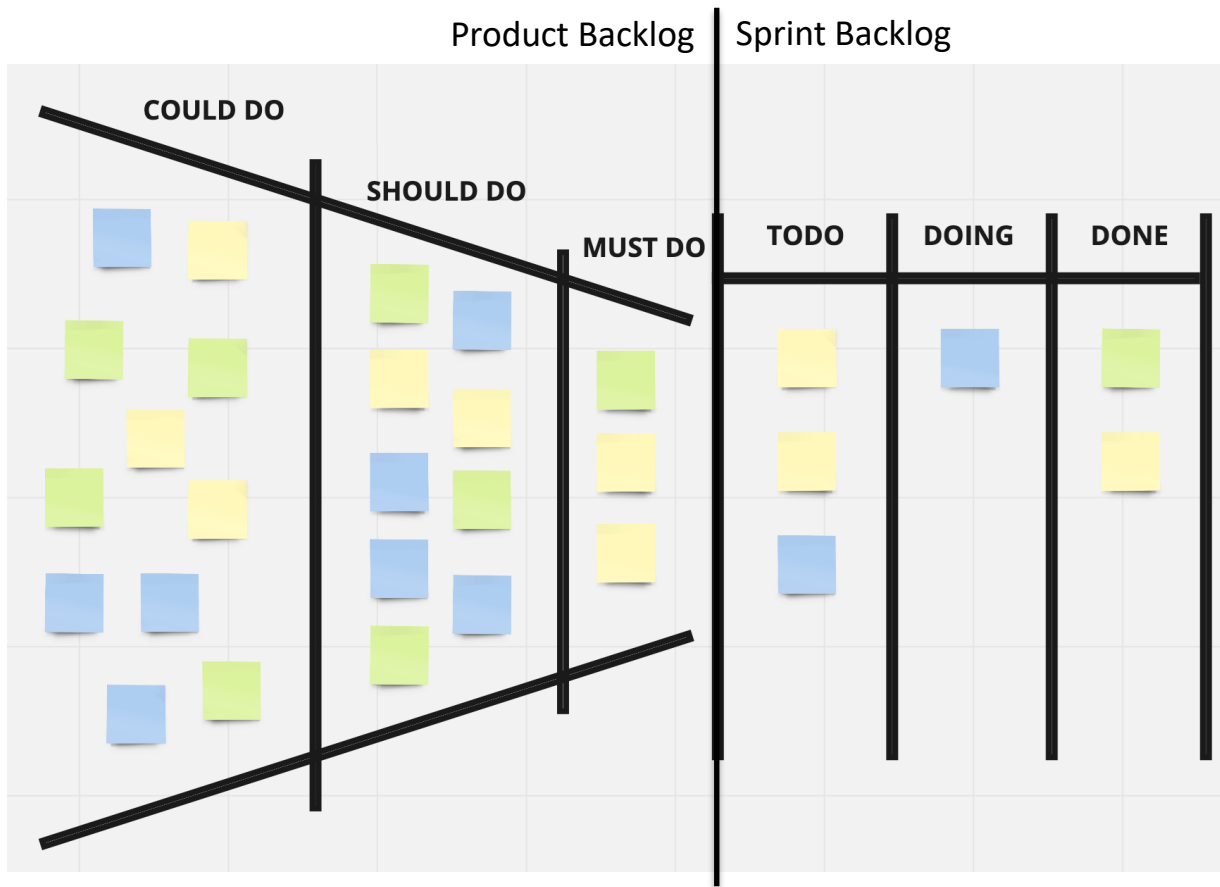Task

Task

Task

Task

Task

Theme vs task
Themes and tasks are similar in that they both help with categorization, adding a sense of order to the work management process. However, themes bring a group of epics or initiatives together under one related banner. Tasks break a story down, creating sub-divisions within an existing section.

Themes are also broader and can spread across the entire company, relating to various epics, stories, and initiatives. Tasks, meanwhile, have a far more specific purpose. They are only used within the context of a single story and are not designed to be shared outside it.
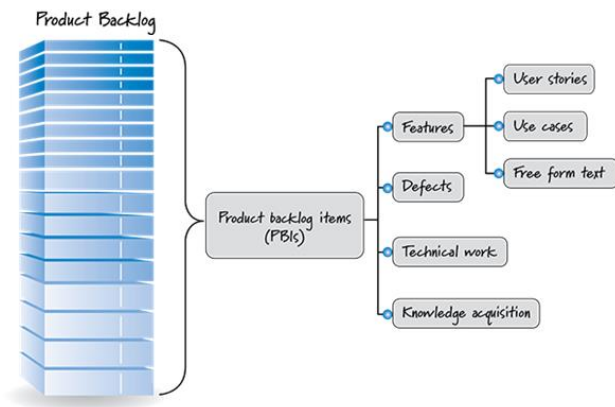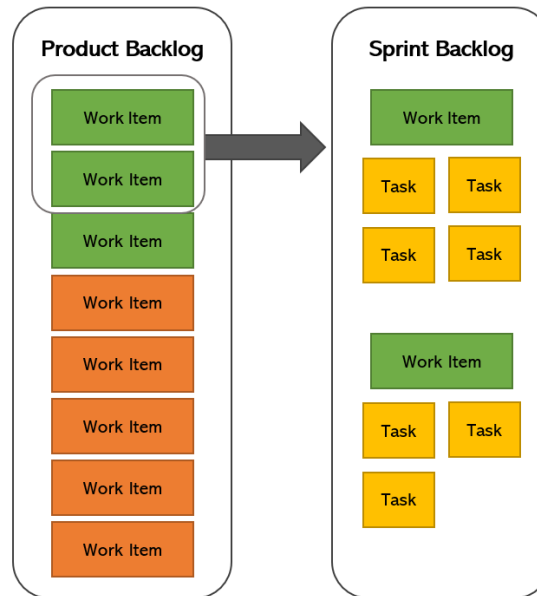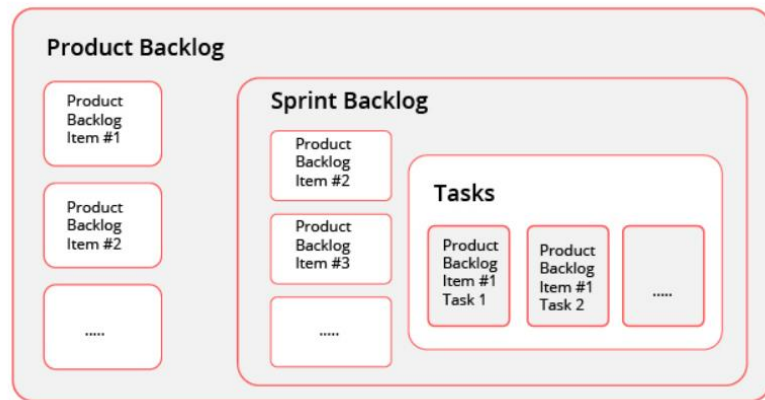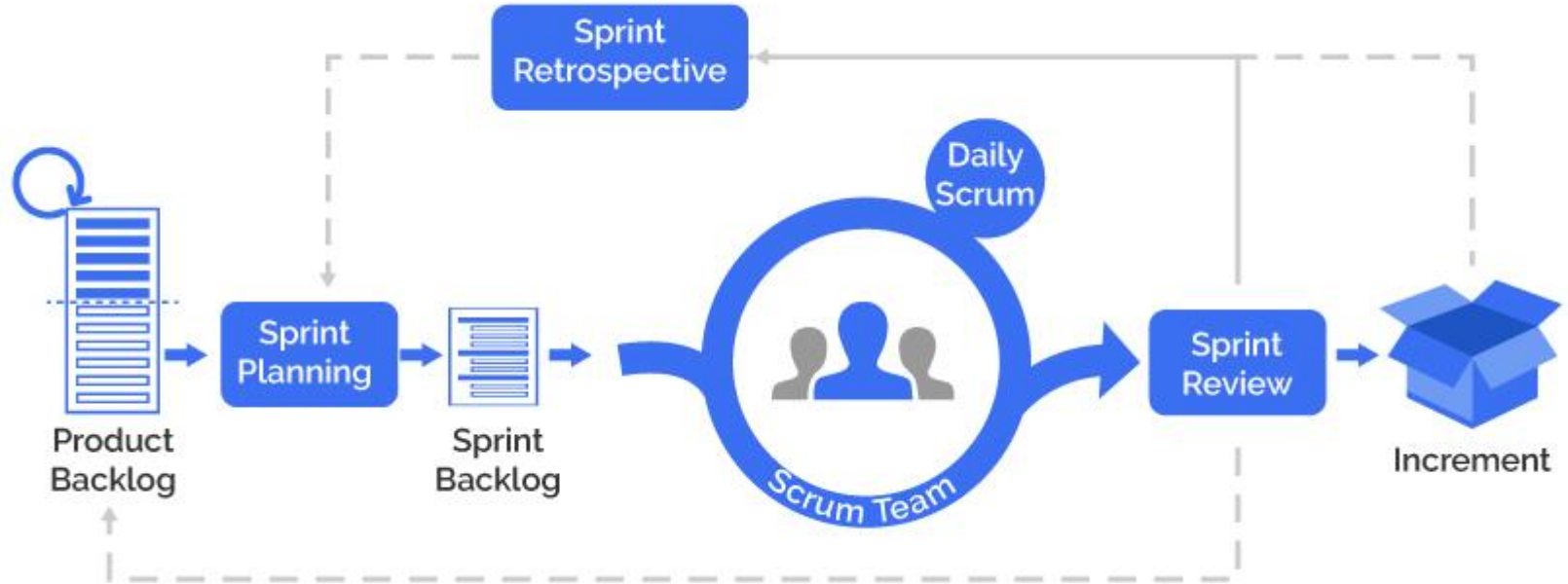
REVATURE

Scrum Process & Basic Ceremonies

Sprint Planning

Daily stand-up

Sprint Review

Sprint Retrospective

Product backlog

Sprint backlog

Sprint

Working product (or) Increment

Sprint Backlog Types

Sprint Backlog Types

Agile Methodology
Scrum Process

Scrum Master
Product Owner
Stakeholders
Users
Team Members

DONE?
DO?
BLOCKERS?
DAILY SCRUM

1. What have I DONE yesterday?
2. What will I DO today?
3. What are my BLOCKERS?

Development
Test/Fix/QA
Manage Sprint
Review Results
Adapt to Change

SPRINT

User Stories

UAT

Burndown Chart

Sprint Planning

Product Backlog → Sprint Backlog

Product Release

Sprint Retrospective

- Product backlog prioritized by business value
- Sprint backlog includes bugs
- Team decides what it can commit to deliver
- Sprint backlog grooming will further prioritize the order of development/implementation

- 2 to 4 weeks of elapsed time
- Iteratively develop/test/review/adapt/manage
- Level of Effort = Story Points
- Velocity = Story Points/Sprint Period
- Dynamically adjust workload
- Test Driven Development
- Weekly Backlog Review

- Product increment – delivery of all Product Backlog items during Sprint
- Release Notes for clients
- Sprint Retrospective
  - What went well?
  - What went wrong?
  - What can we improve?

Understanding AGILE "Story Points"

Story Points are used to estimate the work mentioned in Product Backlog in Agile World.

It's a Unit of measurement and its relative number.

It's the effort that the team will put to develop a user story

Below factors are commonly considered while estimating the story point:

· Amount of Work to be finished

· Risk involved

· Complexity of the work

· Uncertainty involved

The Story point estimation must include all the efforts that will be required to bring the status of the story to DONE i.e. it shall include analysis, coding, testing, review, demo etc.

Example:

· A Socks can be 1 Story Point, a trouser can be 3 Story Points and a blanket can be 5 story point

· A field to be added on a screen can be of 1 Story point but adding exact same 100 fields may not be 100 Story Points

· 1 Non database code can take 2 Story Points to finish but 2 Non Database code may take more than 4 Story points..

Understanding AGILE "Story Points"

There are many techniques that can help you to estimate such as:
· T Shirt size = S M L XL XXL
· Fibonacci Series = 1 2 3 5 8

The actual definition of the estimating numbers can be decided and assigned by the team.
E.g.
S: Its defined as the story which is Quick to Build , have least complexity, its known what needs to be done and it can be developed in a day
M: Its defined as the story which is Quick to Build , have low complexity, its known what needs to be done and it can be developed in 2 days
L: It's defined as the story which is not Quick to Build, have some complexity, somewhat known what needs to be done and it can be developed in 2–3 days
XL: It's defined as the story which is not too Quick to Build, have some complexity, Not sure what needs to be done and it can be developed in 1 week
XXL: It's defined as the story which is not too Quick to Build , have lot of complexity, Not sure what needs to be done and it can be developed in 2–3 weeks

Ideally as it's all about how much a TEAM can achieve, hence the Team itself should discuss and assign Story Points.
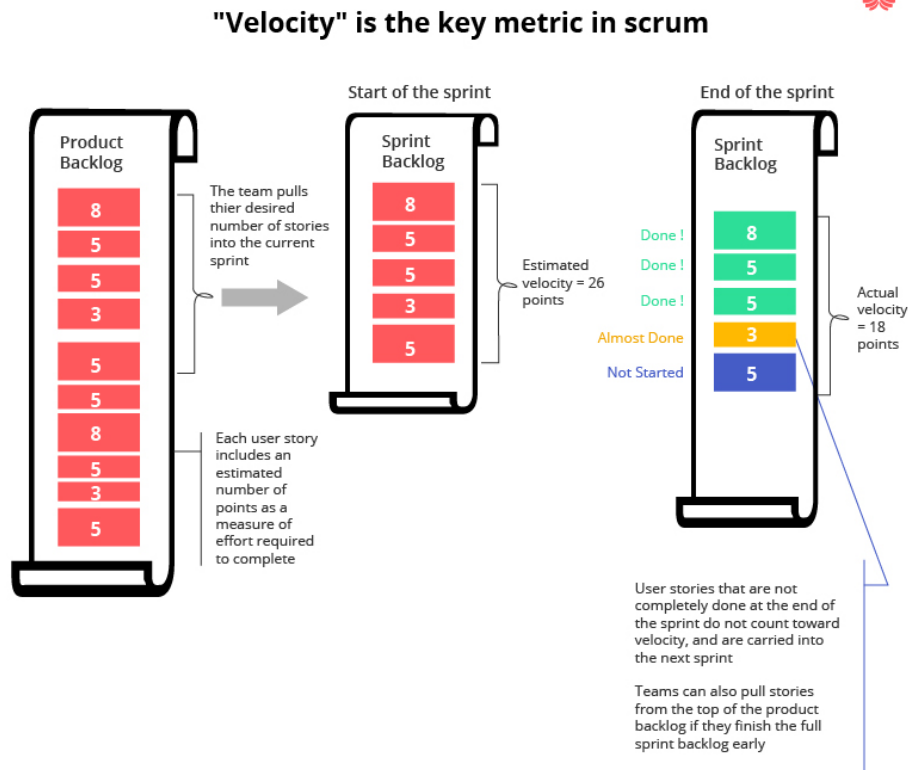
Story Points Matrix

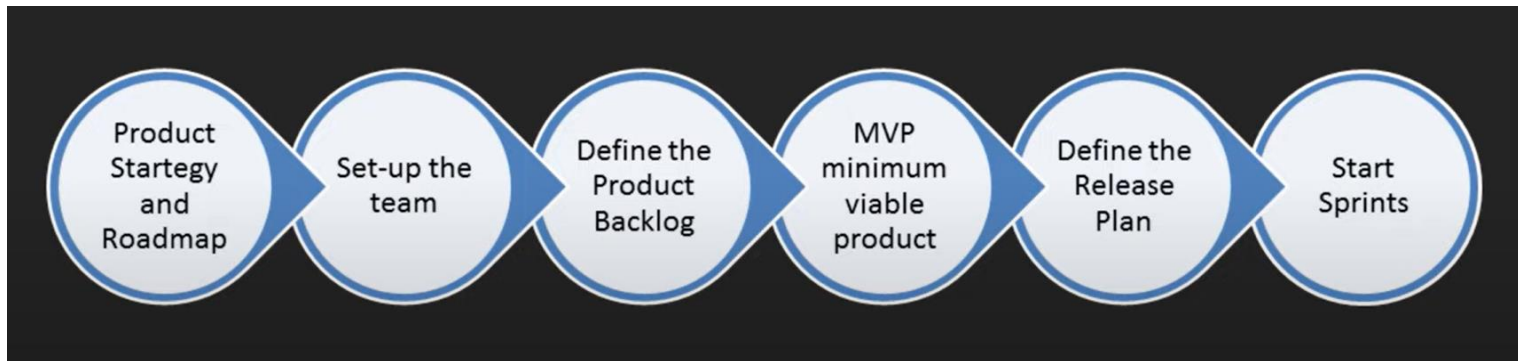| Story Points	Tasks
1	Planting apple seeds
3	Watering the apple trees
5	Harvesting apples from the orchard
8	Making apple cider

Understanding AGILE "Velocity"

Once you collect all the story points you have achieved in a sprint, it shall be called your team's VELOCITY. The Velocity changes in every Sprint but it becomes more predictable after few Sprints and team gets better in knowing amount of work that can be done. The good thing about STORY POINTS is that estimation is relative to items that the team has already finished hence it would mean that our past performance is used as the measure of effort for future work.

The Story Point must be agreed by the team members — that's the essence of it and its most important. Remember — It shouldn't matter who is actually going to implement the user story — a senior resource or a junior resource.

"Velocity" is the key metric in scrum

Product Backlog

| 8 |
| 5 |
| 5 |
| 3 |
| 5 |
| 5 |
| 8 |
| 5 |
| 3 |
| 5 |

The team pulls thier desired number of stories into the current sprint

Each user story includes an estimated number of points as a measure of effort required to complete

Start of the sprint

Sprint Backlog

| 8 |
| 5 |
| 5 |
| 3 |
| 5 |

Estimated velocity = 26 points

End of the sprint

Sprint Backlog

Done !	8
Done !	5
Done !	5
Almost Done	3
Not Started	5

Actual velocity = 18 points

User stories that are not completely done at the end of the sprint do not count toward velocity, and are carried into the next sprint

Teams can also pull stories from the top of the product backlog if they finish the full sprint backlog early

Scrum Advantage & Disadvantages
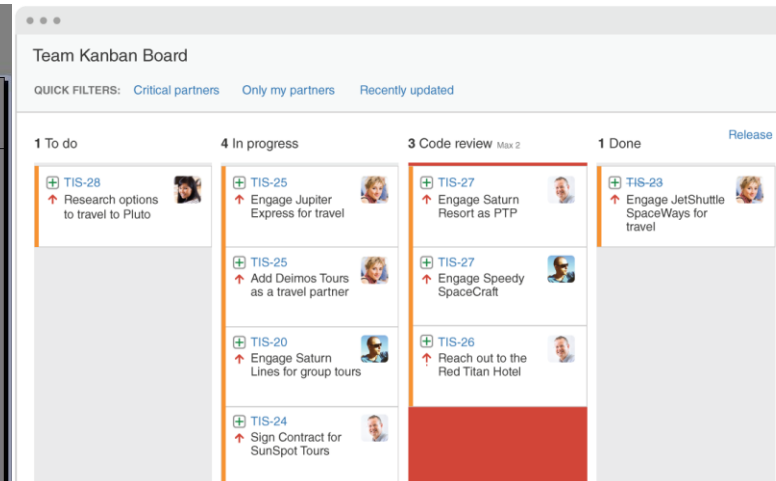
There are some clear advantages to the Scrum approach:
1. They enable quick completion of deliverables
2. They are designed to make effective use of time and money.
3. Projects are divided into easily manageable segments (sprints).
4. They are good for fast-moving projects
5. The provide good awareness among scrum members about what is going on with a project by the entire team (through standups)
6. They are designed well for adopting feedback from stakeholders and customers.

Here's a brief list of problems reported by people using Scrum
1. Because of no clearly defined end-date for an entire project, there is some danger of scope creep.
2. Uncooperative individuals can increase the risk of project failure
3. As sprints do not allow contributing new stories to a sprint unless they are completed, and other sprints cannot begin unless the previous one is completed, this can stall a project.
4. It can be very difficult to implement for large teams
5. It is only successful if everyone is up to par and experienced
6. Daily standups can become irritating for some members, as they can become repetitive
7. It can be difficult to control for quality unless aggressive testing is applied.

Example of a Kanban Board

Backlog	In Progress (3)	Peer Review (3)	In Test (1)	Done	Blocked

Fast Track/ Defect

Team Kanban Board

QUICK FILTERS: Critical partners   Only my partners   Recently updated

1 To do	4 In progress	3 Code review Max 2	1 Done	Release

TIS-28 — Research options to travel to Pluto

TIS-25 — Engage Jupiter Express for travel
TIS-25 — Add Deimos Tours as a travel partner
TIS-20 — Engage Saturn Lines for group tours
TIS-24 — Sign Contract for SunSpot Tours

TIS-27 — Engage Saturn Resort as PTP
TIS-27 — Engage Speedy SpaceCraft
TIS-26 — Reach out to the Red Titan Hotel

TIS-23 — Engage JetShuttle SpaceWays for travel

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

Kanban Advantage & Disadvantages

The advantages of Kanban include the following:
1. It provides a model how work can be accomplished accurately.
2. Standups are driven by the Kanban board. Unlike Scrum standups, they focus specifically on deliverables, and not only individual tasks.
3. Kanban has no need for formal stories, the intent is to keep the flow going.
4. Compared with Scrum, there is less formality/ceremony and overhead.

Some disadvantages reported about Kanban include the following:
1. It is tactical and not strategic and can result in a poor long-range view.
2. Kanban doesn't have a good way of tracking well against goals
3. Kanban has more focus on how tasks should be done rather than how they are actually being accomplished.
4. In reality, it can be difficult to enforce WIP limits.
5. It doesn't work particularly well if teams have remote members, as they cannot see the board (note: this is not difficult to resolve using some excellent online tools).
6. It is poor at breaking down tasks into usable chunks.

Scrumban: Best of both Scrum + Kanban

Scrum is rigid, and very schedule oriented.
Individual sprints must meet the preset timeframe
by the scrum master.
Kanban is not limited into time boxes. It is
iterative, and change occurs evolutionarily.

Scrumban Advantages
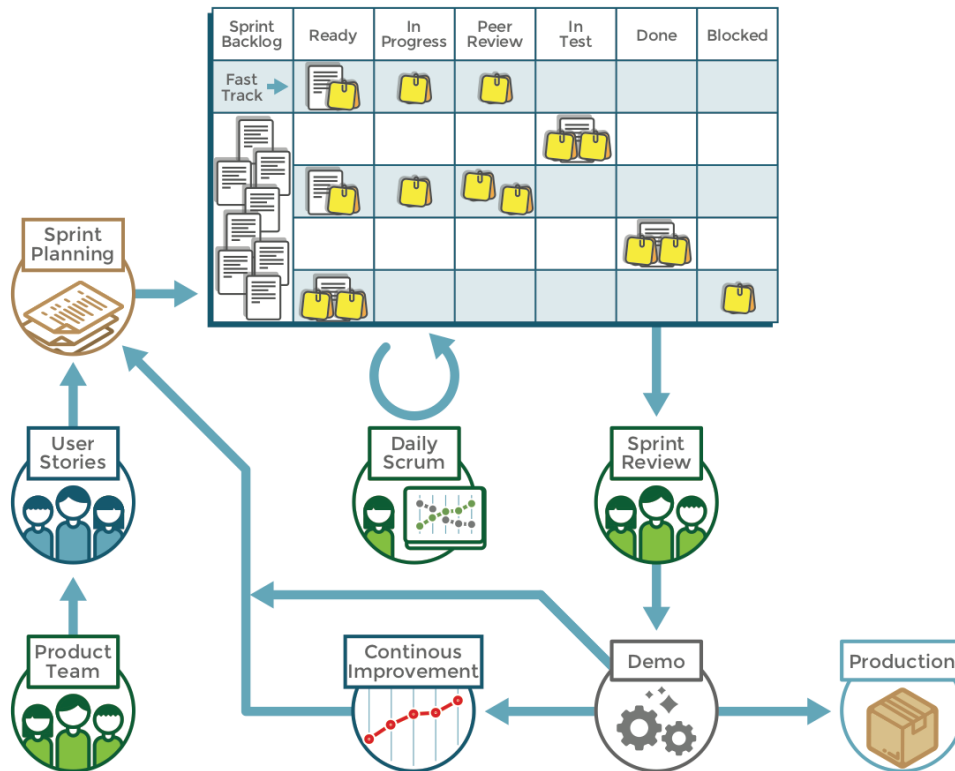
Allows continuous improvement of the workflow by learning from the past.
Whereas Scrum is focused on pure workflow, once a sprint is in place, it's difficult to make changes. However, within Scrumban, if a developer during another sprint notices a problem, not only can he or she modify the current sprint, by adding important pieces into the flow, this knowledge can be shared across other sprints to improve the entire process.

No limits on the size of stories brought into the board.
In Scrumban, stories can be larger than the normal one, two, or five bar size; this allows bringing in stories that may have come from another sprint or epic, even if it wasn't planned.

Ensures continuous flow of work.
Work no longer needs to stop if a developer has completed her part of a sprint. She can move on to the next task without having to wait for a new sprint to begin.

Allow for Specialists
The team can remain cross-functional however there is also space to allow for specialized resources. Sometimes a developer or Ops professional may have skills that are not necessarily shared by others in the group.   Often his work within a sprint may be minimal compared to others, but he has other tasks that need to be accomplished. In Scrumban, is no reason for him to sit on his hands while everyone else finishes their roles. By bringing this person's tasks outside of the individual sprint, he can work in his own stream alongside existing sprints.

eXtreme Programming (XP)

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development.
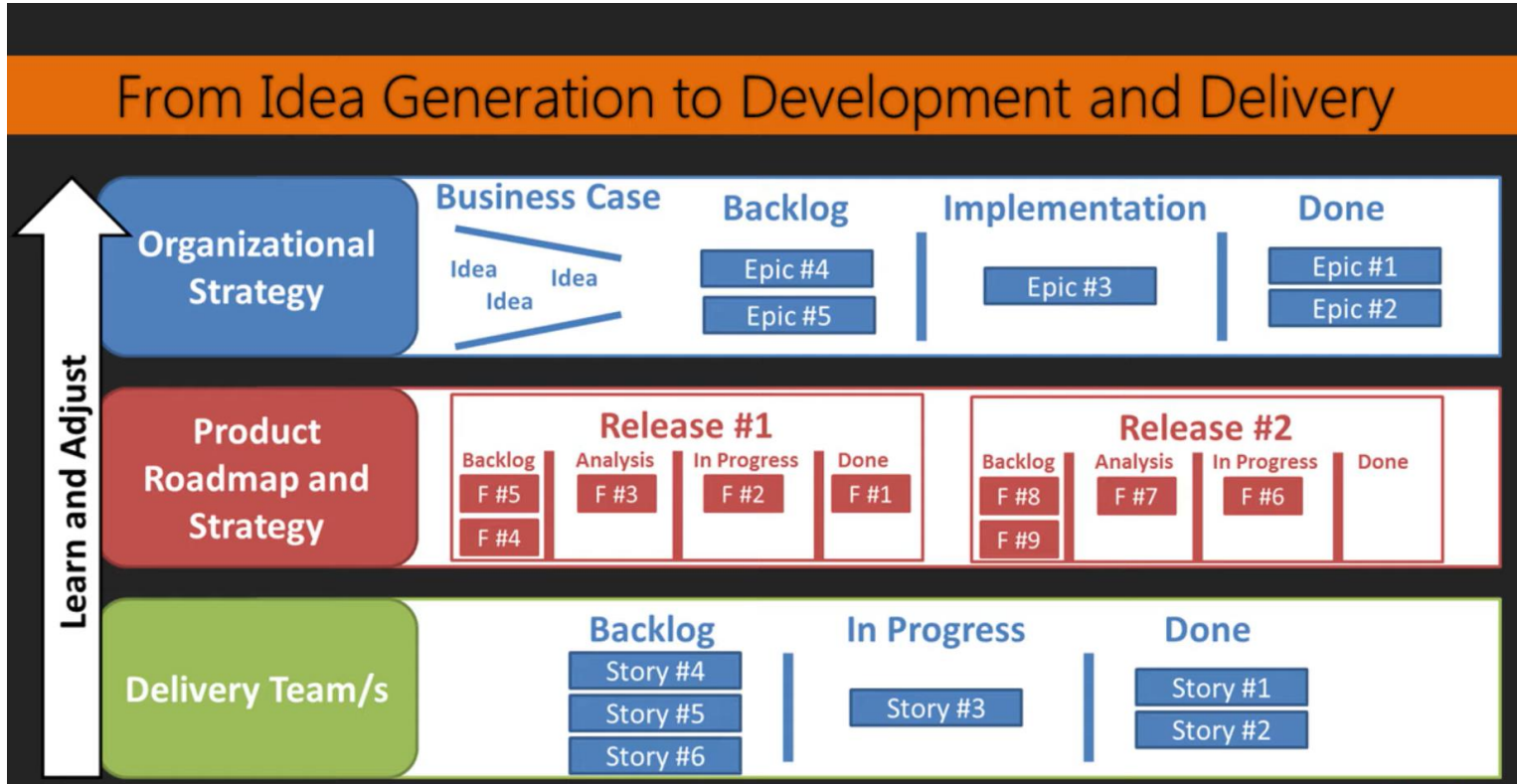
When Applicable

The general characteristics where XP is appropriate were described by Don Wells on www.extremeprogramming.org:

- *Dynamically changing software requirements*
- *Risks caused by fixed time projects using new technology*
- *Small, co-located extended development team*
- *The technology you are using allows for automated unit and functional tests*

The original twelve practices are listed below. If you would like more information about how these practices were originally described, you can visit http://ronjeffries.com/xprog/what-is-extreme-programming/.

1. *The Planning Game*
2. *Small Releases*
3. *Metaphor*
4. *Simple Design*
5. ***Testing***
6. ***Refactoring***
7. ***Pair Programming***
8. *Collective Ownership*
9. ***Continuous Integration***
10. *40-hour week*
11. *On-site Customer*
12. *Coding Standard*

Appendix

Product Roadmap vs Product Backlog

PRODUCT ROADMAP
WHY?

PRODUCT BACKLOG
HOW?

	PRODUCT ROADMAP	PRODUCT BACKLOG
Content	High-level: themes and epics or outcomes and goals	Task-level: user stories and defects
Audience	Executive team (and other stakeholders)	Primarily for product and development teams
Intent	Conveys a strategy	Conveys tactical steps in execution of plan
Time Frame	Varies, typically ~3 months	1 or 2 sprints

The Fibonacci sequence is this: 0,1,1,2,3,5,8,13,21,34... You basically add the previous two numbers to get the next number. It wasn't really "created" per se, but it was more discovered. If you create squares each with the side length of a Fibonacci number, you can create a very unique spiral. Like this-
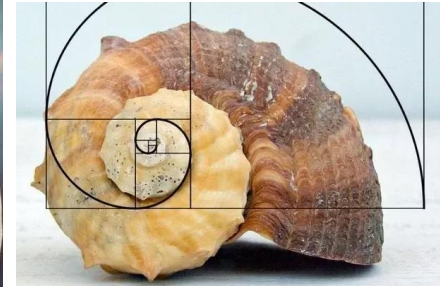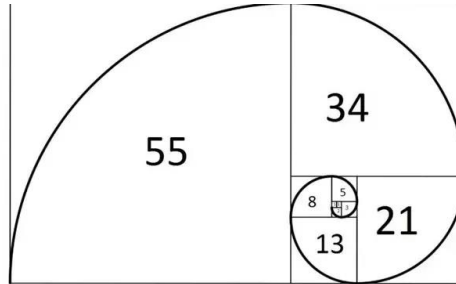
This mysterious sequence appears all around us in nature. The petals of a flower, the seeds of fruits, rows of seeds on a sun flower or the lobes of pinecones and even the spirals on a shell develop or add up to the Fibonacci numbers.

FIBONACCI SEQUENCE

A series of numbers, starting from 0 where every number is the sum of the two numbers preceding it.
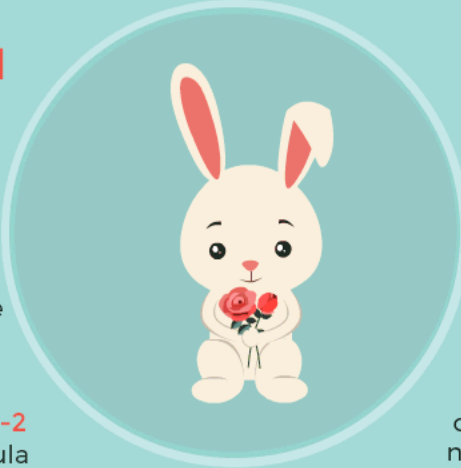
0,1,1,2,3,5,8,13,21,34,55.... and so on

Named after
FIBONACCI
An Italian mathematician

Year 1202
The year it was first introduced to the western world in the book "Liber Abaci"

$x_n = x_{n-1} + x_{n-2}$
Mathematical formula

1.618
"Phi" or the "Golden Ratio" The ratio of any two consequent numbers of the sequence

Nature's code
Because it is observed in several natural phenomena

Copyright © 2016 www.mocomi.com

Fibonacci Sequence Examples

- Presence of Fibonacci numbers in nature is that of a **Sunflower**. It has many Fibonacci spirals contained in it.
- On a **Musical keyboard** , while playing the 8 basic tones (octave). U need exactly 8 keys. Between those 8 keys lie 5 black keys. Further these black keys are divided into 2 sets. One of which contains 2 black keys and the other contains 3. If you observe clearly , all the numbers highlighted are Fibonacci numbers.
- Our **finger print** also contain the famous Fibonacci spiral.
- If we observe the **plant growth** , even there , we can find Fibonacci numbers playing an important role.
- How can we forget the spiral seen in a **shell** . it is one of the most beautiful creation(as per me). It exactly represents a Fibonacci spiral.

https://www.youtube.com/watch?v=ahXIMUkSXX0

REVATURE
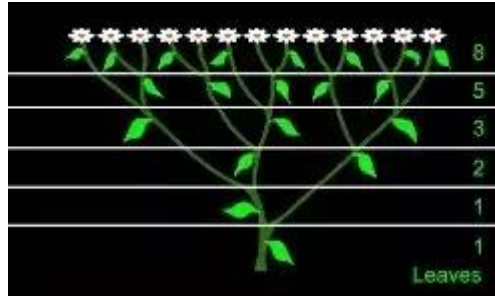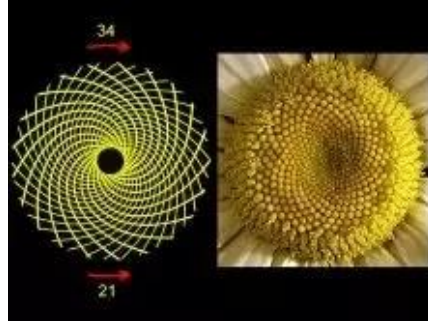
In software engineering (agile practice), the fibonacci sequence is used in estimation for the complexity of a task.

A team member can only choose story points from numbers appearing in the fibonacci sequence.

Suppose we have one User Story/ problem Statement in front of us that need to be resolved but in order to estimate the efforts, tentative time lines, we have to determine the complexity of the problem. Its very difficult to estimate this as we just have very high level information about the problem at this stage. so for doing this initial estimation, we prefer Fibonacci series over normal scale of 1 to 10.

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

How it works, if you feel you have all the relevant information to solve the problem, you can rate the difficulty as 1 but if you feel the problem statement is yet not clear and it will take more time and resources to solve the problem, then you have to start traversing the series from left to right and have to add current number to previous number to estimate the complexity/difficulty of the problem.

You can take these estimated numbers from your entire team for one particular user story/Problem Statement, then you can compare these numbers with them and can have discussion with them to determine final estimated complexity. This estimated complexity of user story/Problem Statement will help you to do initial level resource planning and estimate timelines to resolve this problem.