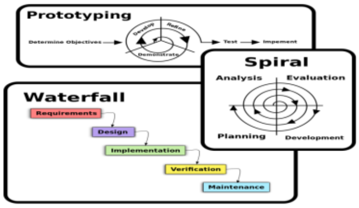
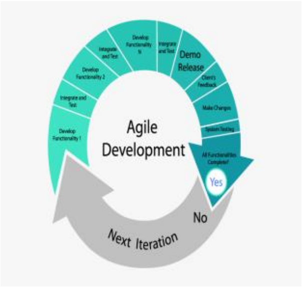
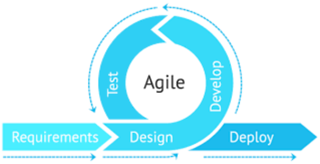


Software Engineering
Agile Development





Outline

- Agility and Cost of Change
- Agile Process
- Introduction to Agile Process Models:
  - Extreme Programming
  - Adaptive Software Development
  - Dynamic Systems Development Method
  - Scrum
  - Crystal
  - Feature Driven Development
  - Lean Software Development
  - Agile Modelling
- Agile Unified Process
- Advantages and Disadvantages of Agile.

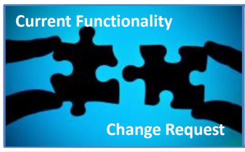
Agility

Agility is **ability to move quickly and easily**.

It is a property consisting of **quickness**, **lightness**, & **ease of movement**.

- The ability to **create** and **respond to change** in order to profit in a unstable global business environment.
- The ability to **quickly reprioritize use of resources** when requirements, technology, and knowledge shift.
- A very **fast response to sudden** market **changes** and emerging threats by intensive customer interaction.
- Use of **evolutionary**, **incremental**, and **iterative delivery** to converge on an optimal customer solution.
- Maximizing **BUSINESS VALUE** with **right sized**, **just- enough**, and **just-in-time processes** and **documentation**.

What is Agility?



Effective response to change

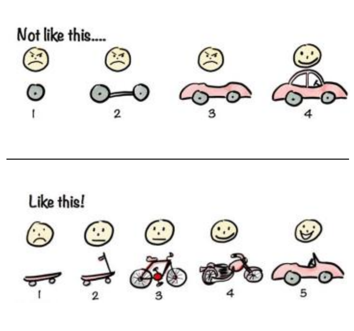
Organizing a team so that it is in control to perform the work

Effective communication among all stakeholders

Drawing the customer onto the team

Eliminate the "us and them" attitude

Rapid and Incremental delivery of



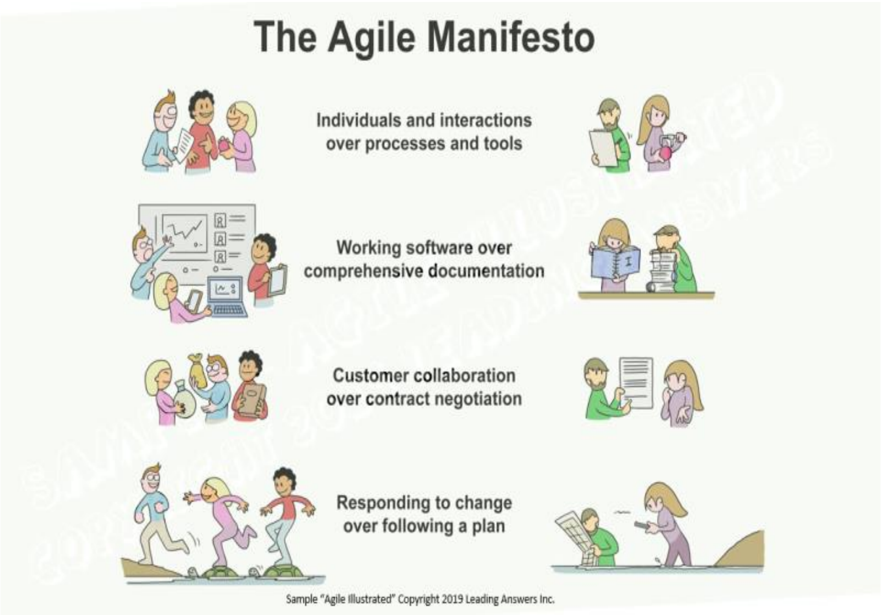
17 software Gurus

Agile manifesto was formally produced by *17 developers* during an outing on *February 11-13, 2001*, at The Lodge at Snowbird ski resort in Utah.

Agile Process

- Agile software process addresses **few assumptions**
  - **Difficulty in predicting changes** of requirements and customer priorities.
  - For many types of software; **design** and **construction** are **interleaved** (mixed).
  - **Analysis**, **design**, **construction** and **testing** are **not** as **predictable** as we might like.
- An agile **process** must **adapt** incrementally.
- To accomplish incremental adaptation, an agile team **requires customer feedback** (so that the appropriate adaptations can be made).

Agile Manifesto



The Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Sample "Agile Illustrated" Copyright 2019 Leading Answers Inc.

Agile Manifesto

- The Agile manifesto is all about giving the preference to below four factors:
  - People
  - Product
  - Communication and
  - Responsiveness
- Agile is not about "you tell me everything you want", which is called requirements gathering; this step can take months to document it. Only after documentation, you will be able to start working on software development, and you will get a working product at the last step. The problem here will be "making a change".
- Agile focuses on exactly what is required by the customer; it does not get into making plans and plots; it just goes with the flow and works on small tasks.

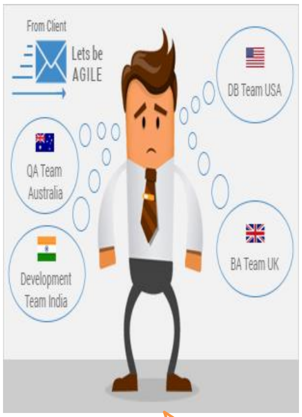
Agile Principles

- **Highest priority** is to **satisfy** the **customer** through early & **continuous delivery** of **software**
- **Welcome changing** requirements
- **Deliver** working **software frequently**
- **Business people** and **developers** must **work together**
- **Build** projects **around motivated** individuals
- Emphasize **face-to-face conversation**
- **Working software** is the **measure** of **progress**
- Continuous **attention** to **technical excellence** and **good design**
- **Simplicity** – the art of maximizing the amount of work done
- The best designs emerge from **self-organizing teams**
- The **team tunes** and **adjusts** its **behaviour** to become more effective
- Agile process promote **sustainable development**.

Where Agile methodology not work

These are the requirements lets meet after 3 months when they are all delivered.
err... But... We follow A..G..I..L..E

From now on we're going to change how we do things here. We need to be Agile.
What does it mean?



Project plan & requirements are clear & unlikely to change

Unclear understanding of Agile Approach among Teams

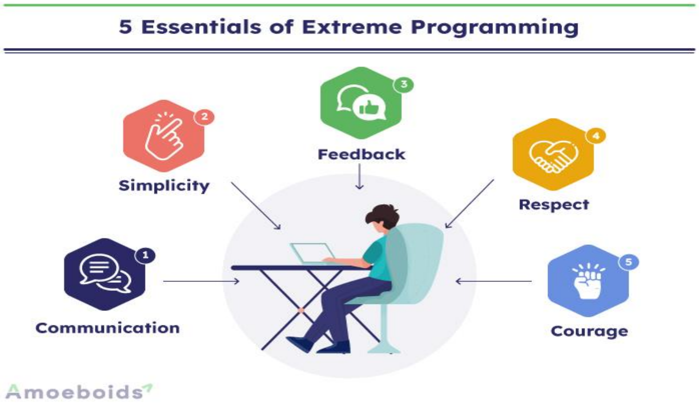
Big Enterprises where team collaboration is tough

Agile Process Models

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)
- Crystal
- Agile Modelling (AM)

Extreme Programming

- The most widely used approach to agile software development
- A variant of XP called **Industrial XP (IXP)** has been proposed to target process for large organizations
- It uses **object oriented approach** as its preferred development model



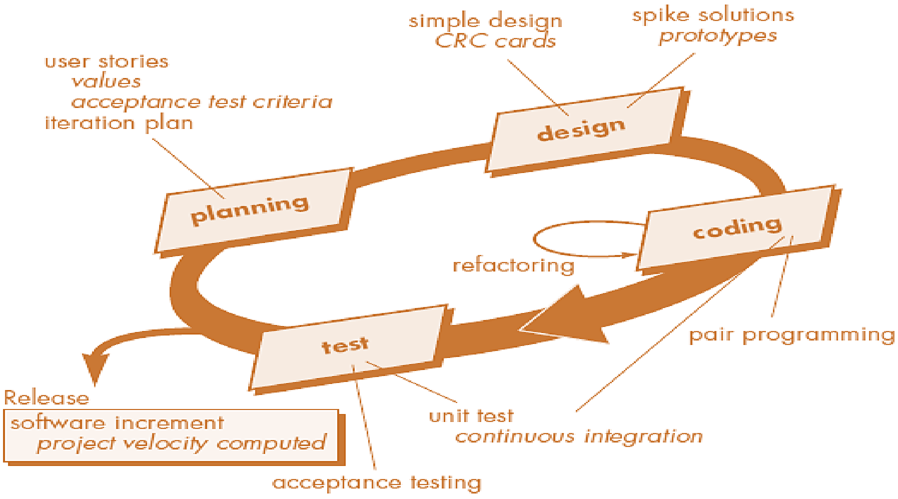
Extreme Programming

XP Values

- **Communication:** To achieve effective communication, it emphasized close & informal (verbal) collaboration between customers and developers
- **Simplicity:** It restricts developers to design for immediate needs not for future needs
- **Feedback:** It is derived from three sources the implemented software, the customer and other software team members, it uses Unit testing as primary testing
- **Courage:** It demands courage (discipline), there is often significant pressure to design for future requirements, XP team must have the discipline (courage) to design for today
- **Respect:** XP team respect among members

The XP Process

- It considers four framework activities
- 1. Planning ◆ 2. Design ◆ 3. Coding ◆ 4. Testing



The XP Process

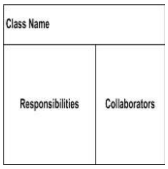
Planning

- ○ User Stories
  - ○ **Customers assigns value** (priority)
  - ○ **Developers assigns cost** (number of development weeks)
- ○ Project velocity
  - ○ Computed at the end of first release
  - ○ **Number of stories implemented in first release**
  - ○ Estimates for future release
  - ○ **Guard against over-commitment**

The XP Process

Design

CRC card



- ○ **Keep-it-Simple** (Design of extra functionality is discouraged)
- ○ **Preparation of CRC card** is work project
  - ○ CRC cards identify and organize object oriented classes
- ○ **Spike Solutions (**in case of difficult design problem is encountered**)**
  - ○ Operational prototype intended to clear confusion
- ○ Refactoring
- ○ Modify internals of code, No observable change

Coding

- ○ **Develops** a series of **Unit test** for stories included in current release
- ○ **Complete code** perform **unit-test** to get immediate feedback
- ○ XP recommend **pair-programming**, **"Two heads are better than one"**
- ○ **Integrate code** with other team members, this **"continuous integration"** helps to avoid compatibility & interfacing problems, **"smoke testing"** environment to uncover errors early

Testing

- ○ **Unit test** by **developers** & fix small problems
- ○ **Acceptance tests** - Specified by **customer**
- ○ This encourages regression testing strategy whenever code is modified

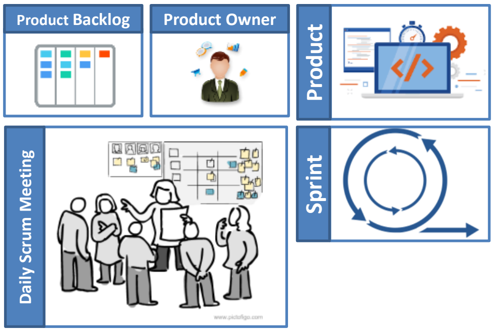
What is Scrum?

Scrum is an agile process model which is used for **developing** the **complex software** systems.

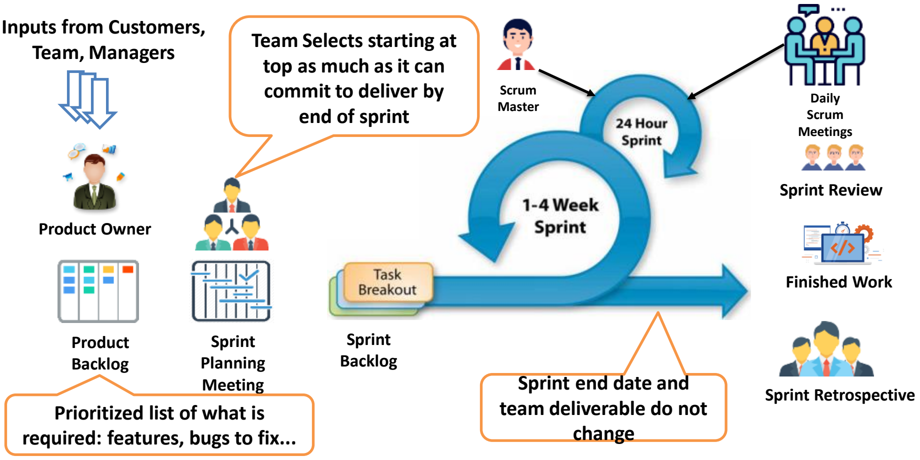
A scrum is a method of restarting play in rugby that involves players packing closely together with their heads down and attempting to gain possession of the ball.

It is a **lightweight process framework**.

Lightweight means the **overhead of the process is kept as small** as possible in order to maximize the productivity.



Scrum framework at a glance



Inputs from Customers, Team, Managers

Team Selects starting at top as much as it can commit to deliver by end of sprint

Scrum Master

Daily Scrum Meetings

24 Hour Sprint

1-4 Week Sprint

Product Owner

Product Backlog

Sprint Planning Meeting

Task Breakout

Sprint Backlog

Sprint Review

Finished Work

Sprint Retrospective

Prioritized list of what is required: features, bugs to fix...

Sprint end date and team deliverable do not change

Scrum

Backlog

- It is a **prioritized list of project requirements** or features that must be provided to the customer.
- The **items can be included** in the backlog at **any time**.
- The **product manager analyses** this **list** and **updates** the **priorities** as per the requirements.

Sprint

- These are the **work units** that are needed **to achieve** the requirements mentioned in the backlogs.

24 Hour Sprint
1-4 Week Sprint
Task Breakout

- Typically the sprints have **fixed duration** or time box (of **2 to 4 weeks, 30 days**).
- **Change** are **not introduced** during the **sprint**.
- Thus sprints allow the team **members** to **work** in **stable** and **short-term environment**

Scrum

Scrum Meetings

- There are **15 minutes daily meetings** to **report** the **completed** activities, **obstacles** and **plan** for **next** activities.
- Following are three questions that are mainly discussed during the meetings.
  - **What** are the **tasks done** since **last meeting ?**
  - **What** are the **issues** that team is **facing ?**
  - **What** are the **next activities** that are **planned?**
- The **scrum master** leads the meeting and **analyses the response** of each team member.
- Scrum meeting **helps** the **team** to **uncover potential problems** as early as possible
- It leads to **“knowledge socialization”** & promotes **“self-organizing team structure”**

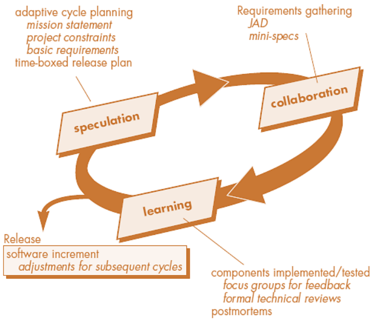
Scrum

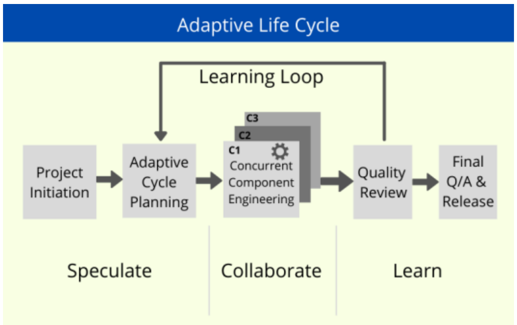
Demo

- Deliver **software increment** to customer
- Implemented functionalities are **demonstrated** to the customer

Adaptive Software development (ASD)

- This is a technique for building complex software systems using iterative approach.
- ASD focus on working in collaboration and team self-organization.
- **ASD incorporates three phases :**
    - 1. Speculation, 2. Collaboration & 3. Learning





Adaptive Software development (ASD)

Speculation

- The adaptive **cycle planning** is **conducted**.
- In this cycle planning mainly three types of information is used
  - Customer's **mission statement**
  - Project **constraints** (Delivery date, budgets etc…)
  - **Basic requirements** of the project

Adaptive Software development (ASD)

Collaboration

- In this, **collaboration** among the **members** of **development team** is a key factor.
- For **successful collaboration** and coordination it is necessary to have following **qualities** in every individual
  - **Assist each other** without resentment (offense)
  - Work **hard**
  - **Posses** the required **skill set**
  - **Communicate problems** and help each other
  - **Criticize without** any **hate**

Adaptive Software development (ASD)

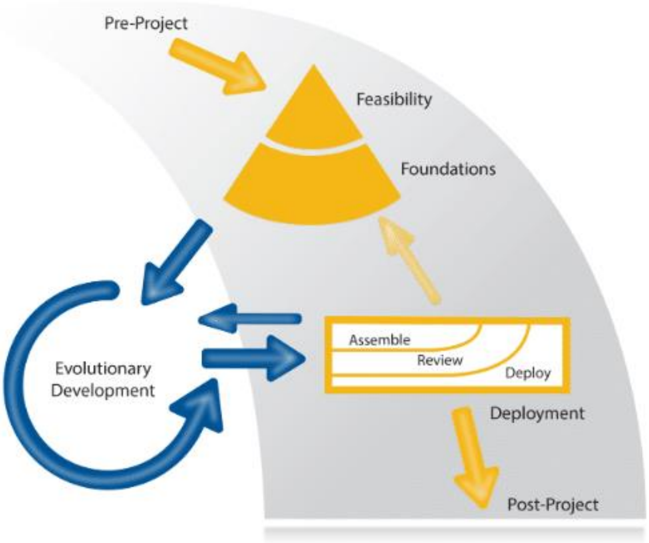
Learning

- Emphasize is on **learning** new **skills** and techniques.
- There are three ways by which the team members learn
  - Focus groups
    - The **feedback** from the **end-users** is obtained.
  - Formal **technical review**
    - This review is conducted for better quality.
  - **Postmortems**
    - Team analyses its own performance and makes appropriate improvements.

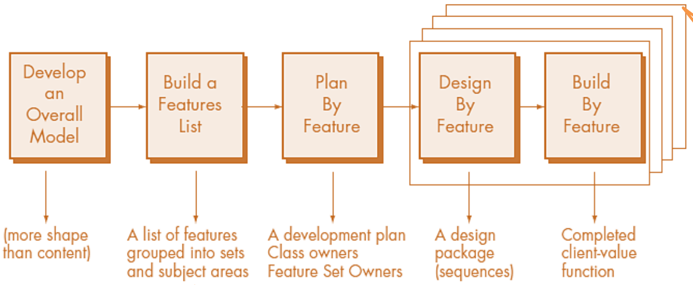
Dynamic Systems Development Methods (DSDM)

- Emphasize is on **learning** new **skills** and techniques.
  - **Feasibility study:** By analysing the business requirements and constraints the viability of the application is determined
  - **Business study:** The functional and informational requirements are identified and then the business value of the application is determined
  - **Functional model iteration:** The incremental approach is adopted for development
  - **Design and build iteration:** If possible design and build activities can be carried out in parallel
  - **Implementation:** The software increment is placed in the working environment

Process of DSDM:



Feature Driven Development(FDD)


Develop an Overall Model
Build a Features List
Plan By Feature
Design By Feature
Build By Feature
(more shape than content)
A list of features grouped into sets and subject areas
A development plan Class owners Feature Set Owners
A design package (sequences)
Completed client-value function

It is practical process model for **object oriented software engineering**

In FDD, the feature means client valued function.

Feature Driven Development(FDD)

1. Develop overall model

- The high-level walkthrough of scope and detailed domain walkthrough are conducted to create overall models.

2. Build feature list

- List of features is created and expressed in the following form
  - **<action>** the **<result> <by for of to>** a(n) **<object>**
  - **For Ex. "Display product-specifications of the product"**

3. Plan by feature

- After completing the feature list the development plan is created

Design by feature

- For each feature the sequence diagram is created

Build by feature

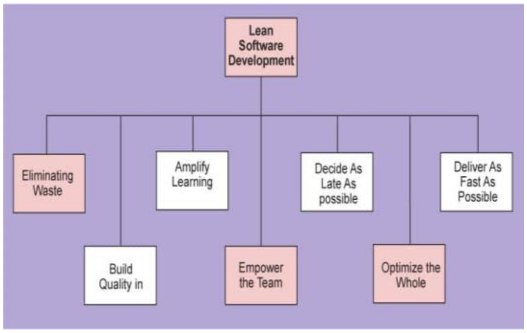
- Finally the class owner develop the actual code for their classes

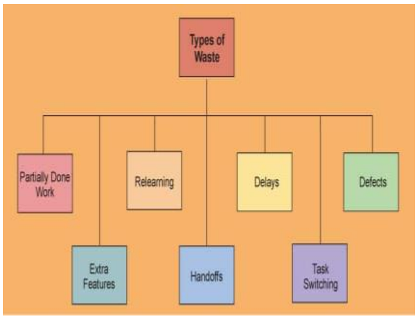
Lean Software Development

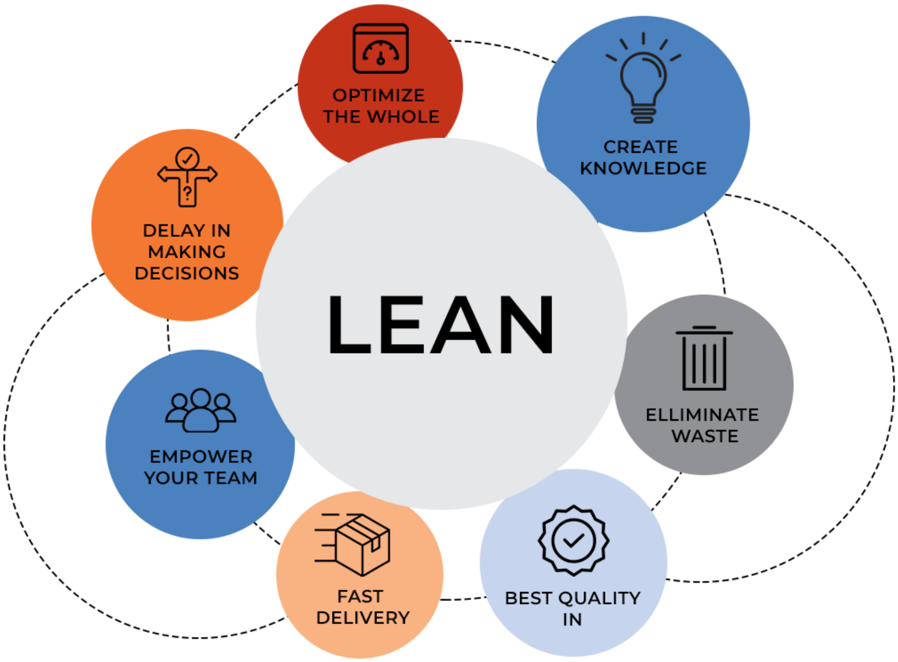
- As articulated in the Digital Project Manager, the focus of the Lean Method is efficiency. Unlike some other Agile methods, this program management methodology is a set of principles, rather than processes, to follow.
- It begins by identifying what is adding value in each situation and then continuously working to emphasize the good while eliminating the bad.
- This method can be applied in project management by examining the way your team processes projects and paring it back to just the essentials.
- The three principles of Lean, according to the Lean Way, were first developed in Japan: Muda (waste), Mura (unevenness), and Muri (overburden), commonly referred to as the 3Ms.

Lean Software Development

- Muda: Eradicate waste. Remove anything that is not adding value to the customer.
- Mura: Eliminate variations. Remove overhead variances and standardize processes.
- Muri: Remove overload. Anything above 60%–70% capacity actually slows work down.



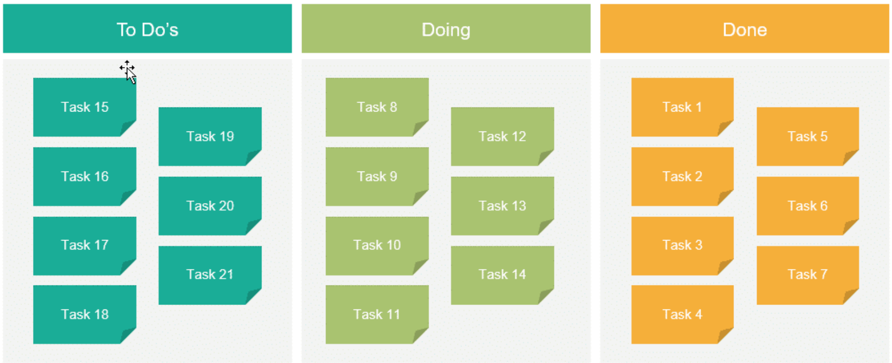




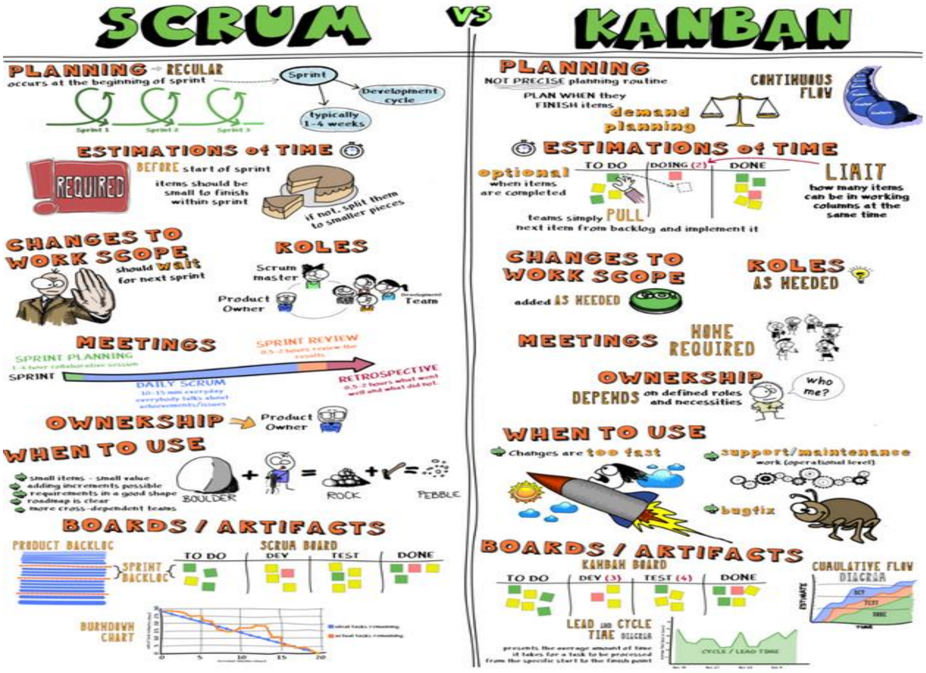
Kanban

- Kanban shares similarities with both Lean and Scrum.
- This project management methodology places an emphasis on efficiency (like Lean) and collaboration (like Scrum).
- Kanban is much less prescriptive in its approach, allowing for greater flexibility and rate of return on deliverables.
- The drive behind Kanban Methodology is to continually release work both faster and of a better quality.
- The core practices of Kanban are:
  - Visualize the workflow.
  - Limit work in progress.
  - Measure the lead time.
  - Make process policies explicit.
  - Continually evaluate improvement opportunities.

Kanban Board consists



To Do's	Doing	Done
Task 15	Task 8	Task 1
Task 16	Task 9	Task 2
Task 17	Task 10	Task 3
Task 18	Task 11	Task 4
Task 19	Task 12	Task 5
Task 20	Task 13	Task 6
Task 21	Task 14	Task 7

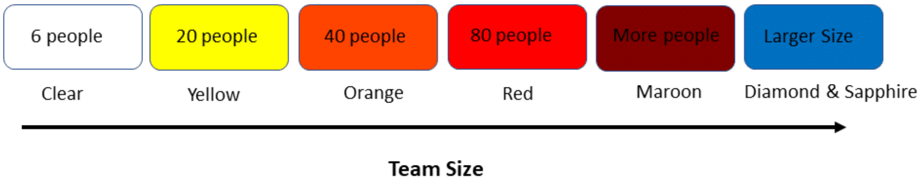

SCRUM
VS
KANBAN
PLANNING → REGULAR
occurs at the beginning of sprint
Sprint
Development cycle
typically 1-4 weeks
Sprint 1
Sprint 2
Sprint 3
ESTIMATIONS of TIME
REQUIRED
BEFORE start of sprint
items should be small to finish within sprint
if not, split them to smaller pieces
CHANGES TO WORK SCOPE
should wait for next sprint
ROLES
Scrum master
Product Owner
Development Team
MEETINGS
SPRINT PLANNING
1-4 hour collaboration session
SPRINT
SPRINT REVIEW
0.5-2 hours review the results
DAILY SCRUM
10-15 min everyday
everybody talks about achievements/issues
RETROSPECTIVE
0.5-2 hours what went well and what did not
OWNERSHIP → Product Owner
WHEN TO USE
small items - small value
adding increments possible
requirements in a good shape
roadmap is clear
more cross-dependent teams
BOULDER
ROCK
PEBBLE
BOARDS / ARTIFACTS
PRODUCT BACKLOG
SPRINT BACKLOG
SCRUM BOARD
TO DO
DEV
TEST
DONE
BURNDOWN CHART
0
5
10
15
20
PLANNING
NOT PRECISE planning routine
PLAN WHEN they FINISH items
demand planning
CONTINUOUS FLOW
ESTIMATIONS of TIME
optional when items are completed
TO DO
DOING (2)
DONE
LIMIT
how many items can be in working columns at the same time
teams simply PULL next item from backlog and implement it
CHANGES TO WORK SCOPE
added AS NEEDED
ROLES AS NEEDED
MEETINGS NONE REQUIRED
OWNERSHIP
DEPENDS on defined roles and necessities
Who me?
WHEN TO USE
Changes are too fast
support/maintenance work (operational level)
bugfix
BOARDS / ARTIFACTS
KANBAN BOARD
TO DO
DEV (3)
TEST (4)
DONE
CUMULATIVE FLOW DIAGRAM
ESTIMATE
DEV
TEST
DONE
TIME
LEAD AND CYCLE TIME DIAGRAM
presents the average amount of time it takes for a task to be processed from the specific start to the finish point
CYCLE / LEAD TIME

Crystal

- Crystal methodology is one of the most lightweight and flexible approaches to develop software.
- The basis of the Crystal Method is on **two critical assumptions:**
  - *First, the team can make itself more efficient by streamlining their work and the project.*
  - *Second, every project is different from others and requires some specific methods and strategies.*
- Crystal methods focus on:-
  - People involved
  - Interaction between the teams
  - Community
  - Skills of people involved
  - Their Talents
  - Communication between all the teams

Crystal

- Crystal method depends on three dimensions:
  - First, Team size
  - Second, Criticality
  - Third, the priority of the project



6 people, Clear

20 people, Yellow

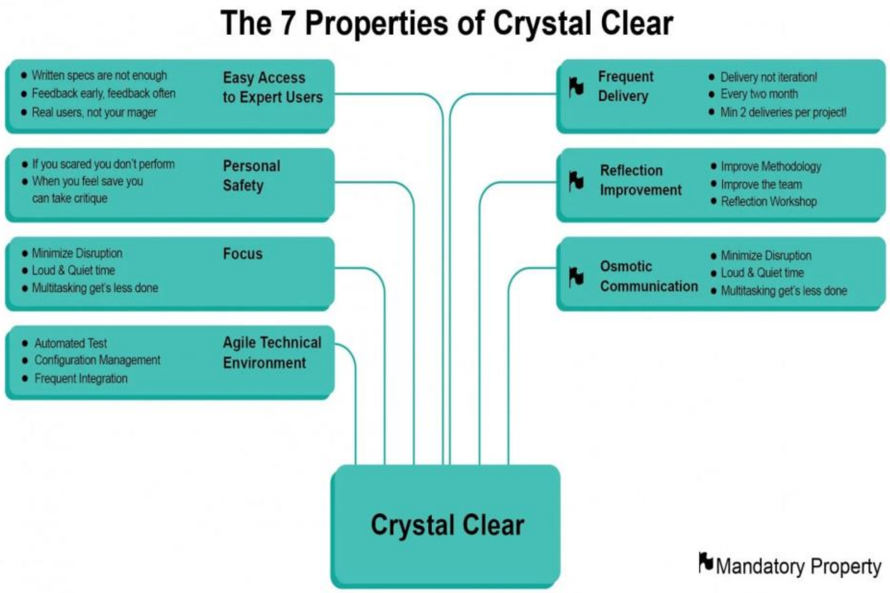
40 people, Orange

80 people, Red

More people, Maroon

Larger Size, Diamond & Sapphire

Team Size



The 7 Properties of Crystal Clear

Easy Access to Expert Users
- Written specs are not enough
- Feedback early, feedback often
- Real users, not your mager

Personal Safety
- If you scared you don't perform
- When you feel save you can take critique

Focus
- Minimize Disruption
- Loud & Quiet time
- Multitasking get's less done

Agile Technical Environment
- Automated Test
- Configuration Management
- Frequent Integration

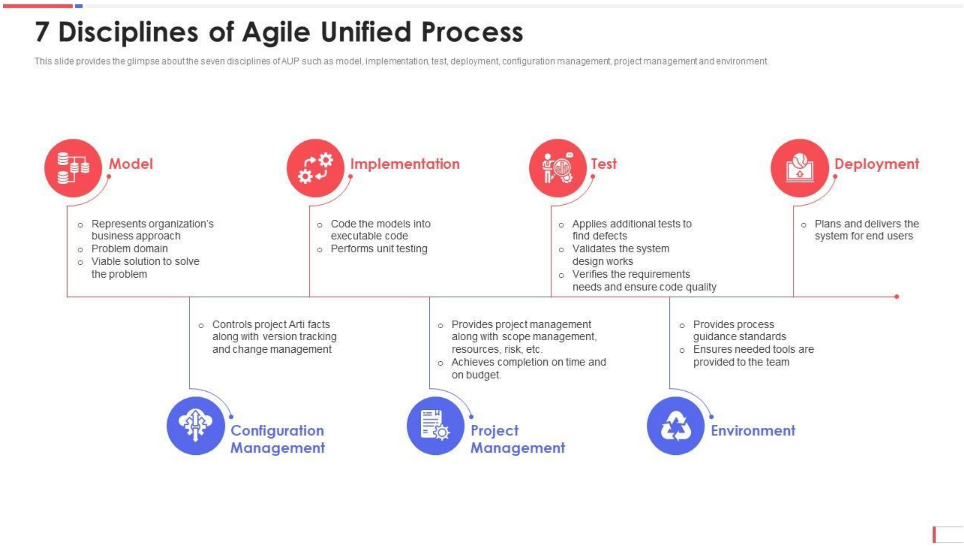
Frequent Delivery ⚑
- Delivery not iteration!
- Every two month
- Min 2 deliveries per project!

Reflection Improvement ⚑
- Improve Methodology
- Improve the team
- Reflection Workshop

Osmotic Communication ⚑
- Minimize Disruption
- Loud & Quiet time
- Multitasking get's less done

Crystal Clear

⚑ Mandatory Property



7 Disciplines of Agile Unified Process

This slide provides the glimpse about the seven disciplines of AUP such as model, implementation, test, deployment, configuration management, project management and environment.

Model

- Represents organization's business approach
- Problem domain
- Viable solution to solve the problem

Implementation

- Code the models into executable code
- Performs unit testing

Test

- Applies additional tests to find defects
- Validates the system design works
- Verifies the requirements needs and ensure code quality

Deployment

- Plans and delivers the system for end users

Configuration Management

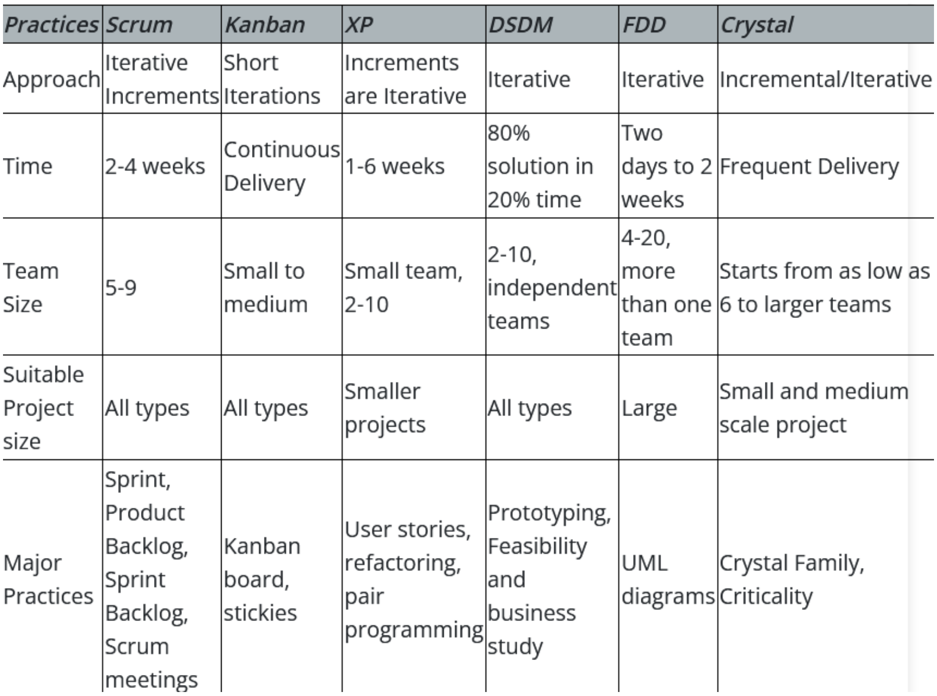
- Controls project Arti facts along with version tracking and change management

Project Management

- Provides project management along with scope management, resources, risk, etc.
- Achieves completion on time and on budget.

Environment

- Provides process guidance standards
- Ensures needed tools are provided to the team



Practices	*Scrum*	*Kanban*	*XP*	*DSDM*	*FDD*	*Crystal*
Approach	Iterative Increments	Short Iterations	Increments are Iterative	Iterative	Iterative	Incremental/Iterative
Time	2-4 weeks	Continuous Delivery	1-6 weeks	80% solution in 20% time	Two days to 2 weeks	Frequent Delivery
Team Size	5-9	Small to medium	Small team, 2-10	2-10, independent teams	4-20, more than one team	Starts from as low as 6 to larger teams
Suitable Project size	All types	All types	Smaller projects	All types	Large	Small and medium scale project
Major Practices	Sprint, Product Backlog, Sprint Backlog, Scrum meetings	Kanban board, stickies	User stories, refactoring, pair programming	Prototyping, Feasibility and business study	UML diagrams	Crystal Family, Criticality

Advantages of Agile Methodology

- Agile is very much suited for projects where requirements and the end product is not very clear.
- It promotes customer satisfaction as their feedbacks and changes are embraced.
- It reduces risk factors as early deliverables are made visible to the end-users.
- Exhaustive planning is not required at the beginning of the development process.
- It is easy to manage with minimal rules and more flexibility.
- Dividing the project into incremental deliverable builds leads to more focus on the quality of the product.

Disadvantages of Agile Methodology

- As it is highly customer-centric, so it can pose a problem when the customer does not have a clear understanding of the product and process.
- Lack of formal documentation and designing leads to a very high dependency on individuals for training and other tasks.
- For complex projects, the resource requirement and effort are difficult to estimate.
- Frequent deliverables, feedback, and collaboration can be very demanding for some customers.
- Because of the ever-evolving features, there is always a risk of the ever-lasting project.

Difference between Agile & Waterfall Methodology

Waterfall Methodology	Agile Methodology
Waterfall involves a large team size where coordination among teams decreases.	Agile intends a smaller team size for higher coordination.
The customer intervenes only after completing the development process.	Continuous feedback is taken from the customer to deliver robust and high-quality products.
Its methodology is quite sequential.	Agile methodology is incremental and iterative.
None of the testing or development levels overlap each other.	The testing and development levels often overlap each other

Difference between Agile & Waterfall Methodology

Waterfall Methodology	Agile Methodology
Acceptance tests are carried out only once, at the last stage.	Acceptance tests are taken continuously after every iteration.
Changes in deliverables are costly.	Changes are obvious so it doesn't impact that much on the deliverable.
Testing is performed towards the end of the project. You don't know whether the deliverable works when you are about to finish it. Any failure in any of the functionality takes you back to the beginning of the product from where it started to find the root cause of it.	Testing is done parallelly in pieces so that if any functionality fails it can be analyzed quickly and easily rectified.

THANKS