## Outline

- What is Agility?
- Agile Process
- Agility Principles
- Where agile methodology not work?
- Agile Process Models
  - Extreme Programming (XP)
  - Adaptive Software Development (ASD)
  - Dynamic Systems Development Method (DSDM)
  - Scrum
  - Feature Driven Development (FDD)
  - Crystal
  - Agile Modelling (AM)

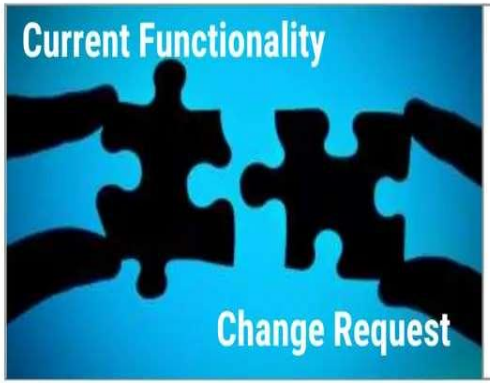
# Agility

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? Cont.




Effective response to change

Effective communication among all stakeholders

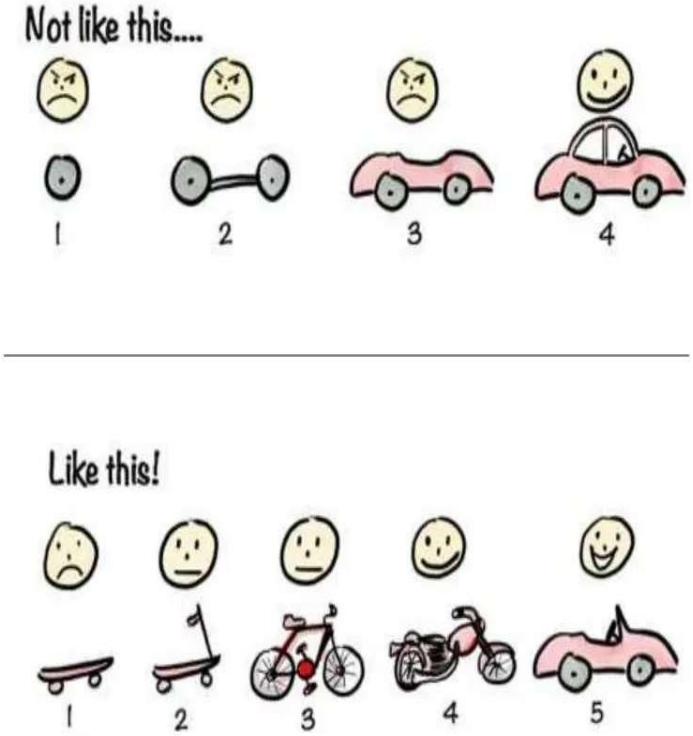
Organizing a team so that it is in control to perform the work

Development Team


Drawing the customer onto the team

Eliminate the "us and them" attitude

Rapid and Incremental delivery of software



# Agile Process

- Agile software process addresses **few assumptions**

> **Difficulty in predicting changes** of requirements and customer priorities.

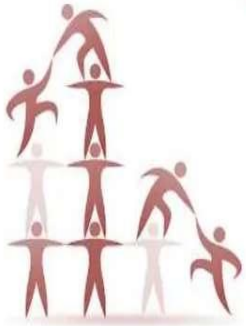
> For many types of software; **design** and **construction** are **interleaved** (mixed).

> **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).

# Agility 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

# Where agile methodology not work

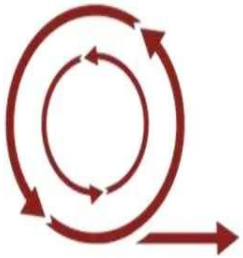

Project plan & requirements are clear & unlikely to change


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 (XP)

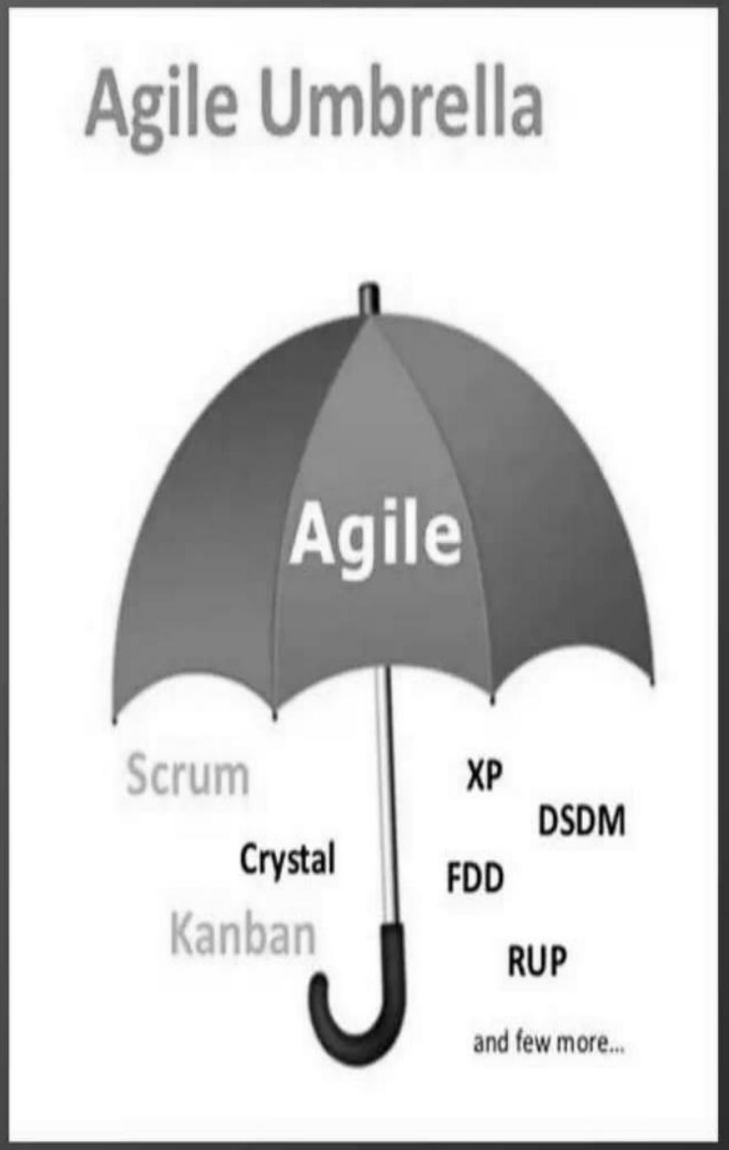
- The most widely used approach to agile software development
- A variant of XP called **Industrial XP (IXP)** has been proposed to target process for large organizations
- It uses **object oriented approach** as its preferred development model

## 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**

# Extreme Programming (XP)

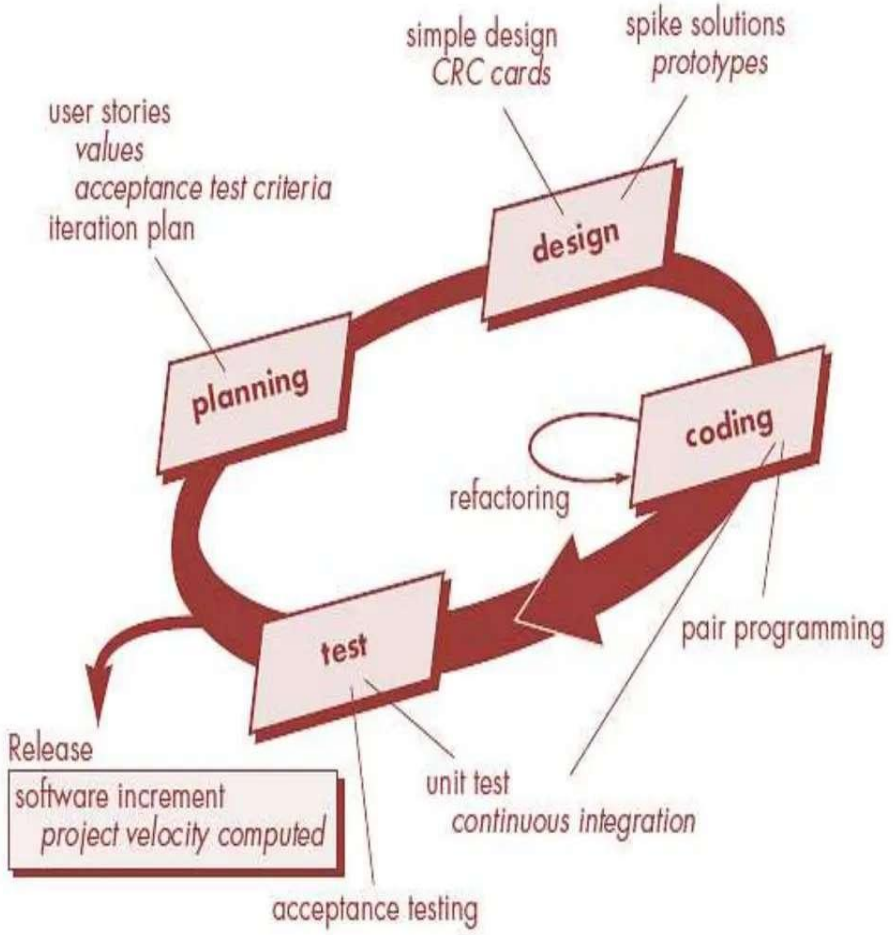
Fun way to develop a software



# The XP Process

It considers four framework activities

**1.** Planning ◆ **2.** Design ◆ **3.** Coding ◆ **4.** Testing




## Planning

### User Stories

- **Customers assigns value** (priority)
- **Developers assigns cost** (number of development weeks)

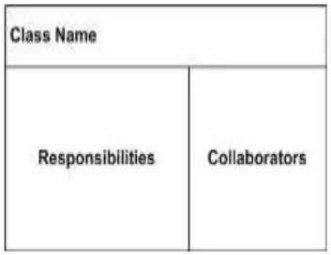
### Project velocity

- 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 cont.

## 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

# Planning Phase

- The main planning process

- Is a meeting that occurs once per iteration

- It is too quickly determine the scope of the next release.

# Release Planning vs Iteration Planning

*Release Planning*: Determining what requirements are included in which near-term releases.

*Iteration Planning*: Plans the activities and tasks.

# Short Releases

- Deploy into production quickly.
- Release new versions in very short cycles.
- As small as possible.

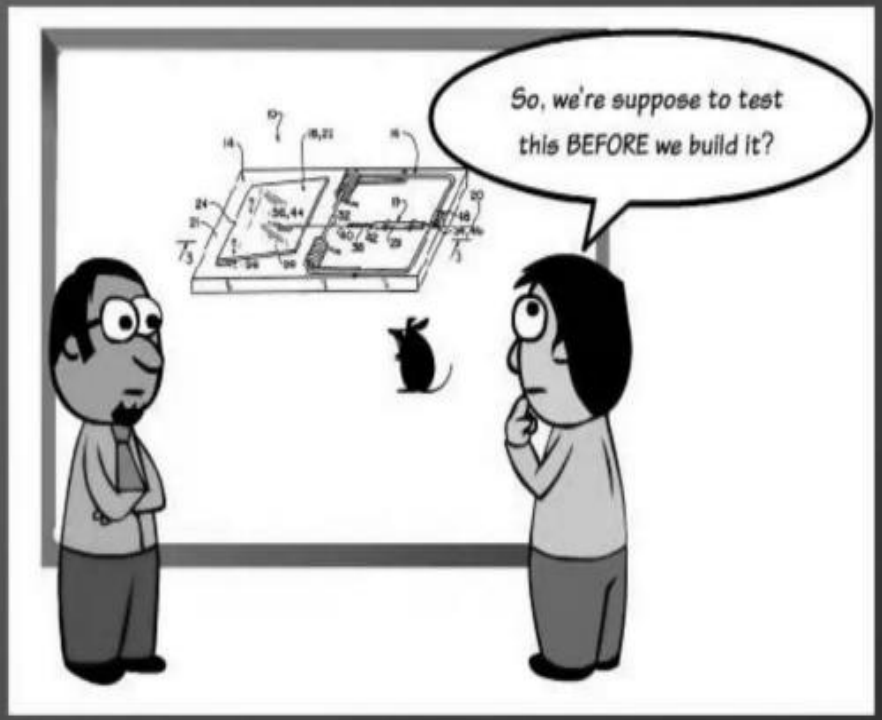
# Short Releases Advantages

## Advantage

- Achievable in a short cycle
- Contains the most valuable requirements
- Frequent feedback
- A working system

# Testing (Test Driven Development)

- Unit testing is done before coding.
- After writing unit tests, pass for the development to continue.
- The result is a system that is capable of accepting change.



# Pair Programming

- All code is written by two programmers sitting at one machine.
- Pairing is dynamic.
- The driver has control of the keyboard and mouse and creates the implementation.
- The navigator watches the driver's implementation .

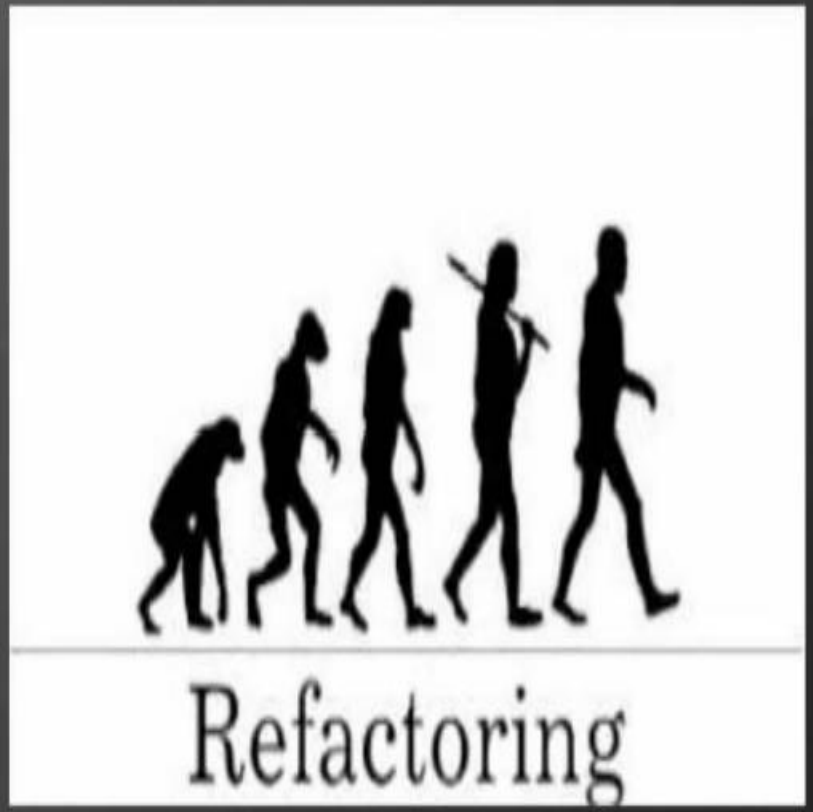
# Pair Programming Advantages

## ADVANTAGES

- Pairs complete their tasks faster .
- Pairs produce higher quality code .
- Pairs are more focused towards their work .
- Pairs feel more confident in their work and they can answer any questions related to their approach .

# Refactoring

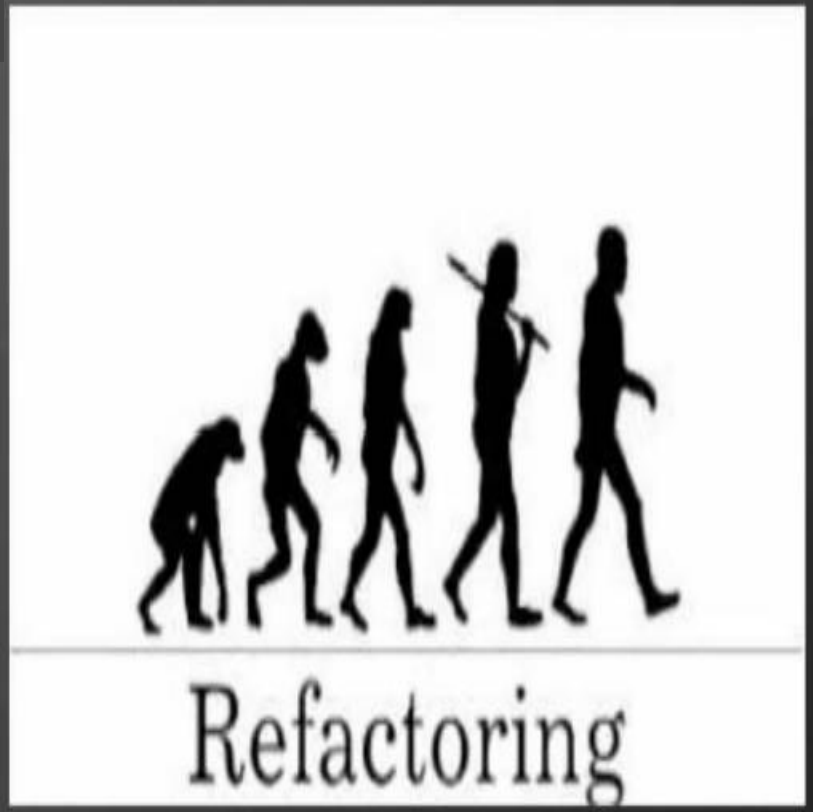
- Refactoring: “Improving the design of Existing code”.
- Removal of Duplication.
- Increases ”Cohesion” of the code and lowers the “coupling”.
- XP practices support each other: They are stronger together than separately.



# Refactoring Advantages

## Advantages:

- Increases the developer knowledge of the system.
- Helps developer to improve the whole product proactively.



# Collective Ownership

- In Extreme Programming, the entire team takes responsibility for the whole of the system. Not everyone knows every part equally well, although everyone knows something about every part.
- Any engineer can modify any code to produce better quality.
- The entire team takes responsibility of the whole system.

# Collective Ownership Advantages

## ADVANTAGE

Promotes the developers to take responsibility for the system as a whole rather than parts of the system.

# Continuous Integration

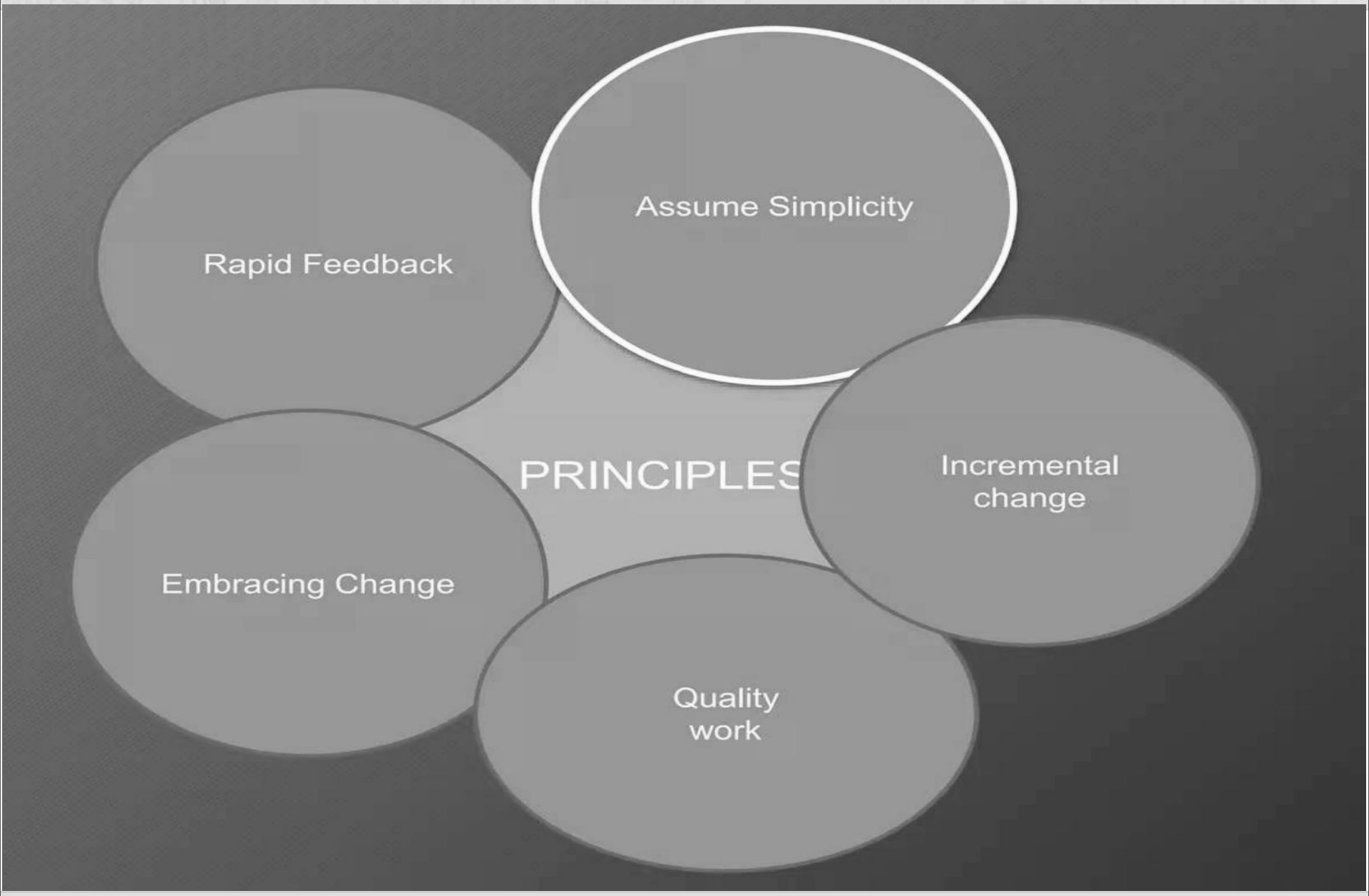
- The unit tests have to run 100% both before and after integration.
- A simple way to do this is to have a machine dedicated to integration.
- Runs the tests until they pass (100% correct).
- Continuous integration should happen more frequently (once or twice a day) .

# Continuous Integration Advantages

## ADVANTAGES

- Reduces the duration , which is otherwise lengthy .
- Enables the short releases practice as the time required before release is minimal .

# XP Principles



# Extreme Programming: Conclusion

## CONCLUSION

### Success in Industry

Rapid development.
Immediate responsiveness to the customer's changing requirements.
Focus on low defect rates.
System returning constant and consistent value to the customer.
High customer satisfaction.
Reduced costs.
Team cohesion and employee satisfaction.

# 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.

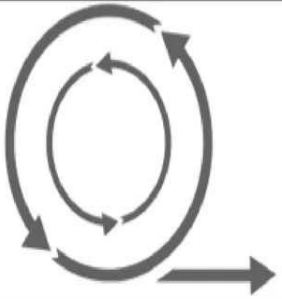
**Product Backlog**

**Product Owner**

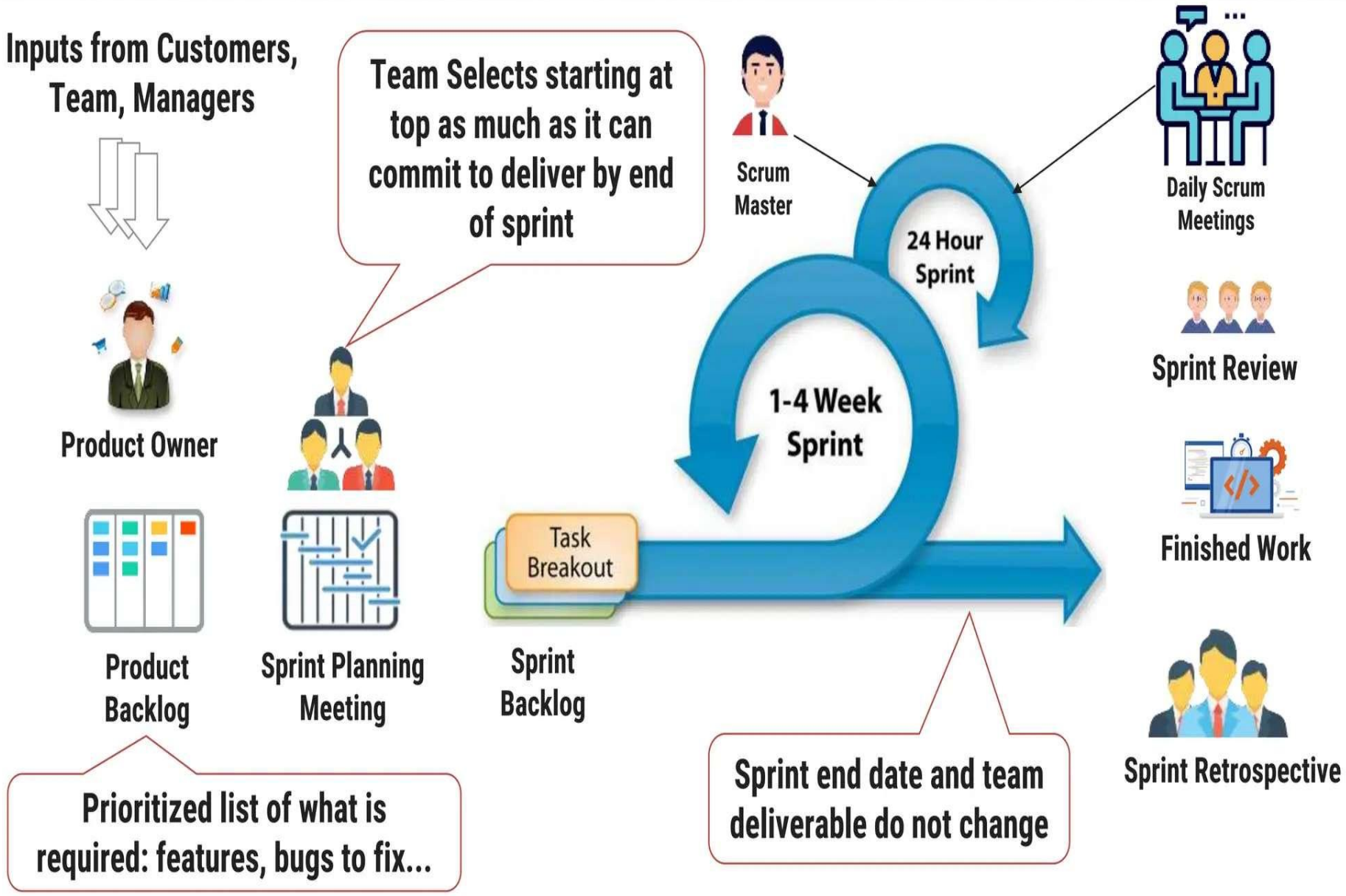
**Product**

**Daily Scrum Meeting**


**Sprint**



# Scrum framework at a glance



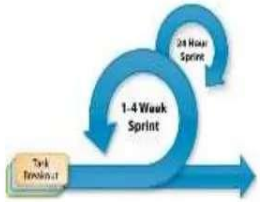
# Scrum cont.

## 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.
- 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 cont.

## 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.
  1. **What** are the **tasks done** since **last meeting ?**
  2. **What** are the **issues** that team is **facing ?**
  3. **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"**

## Demo

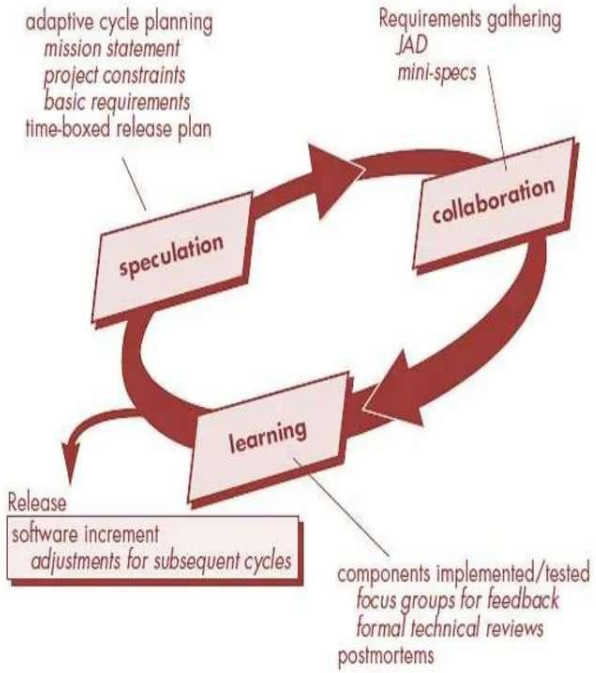
- Deliver **software increment** to customer
- Implemented functionalities are **demonstrated** to the customer

# 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




## 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) cont.

## 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**

## 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.

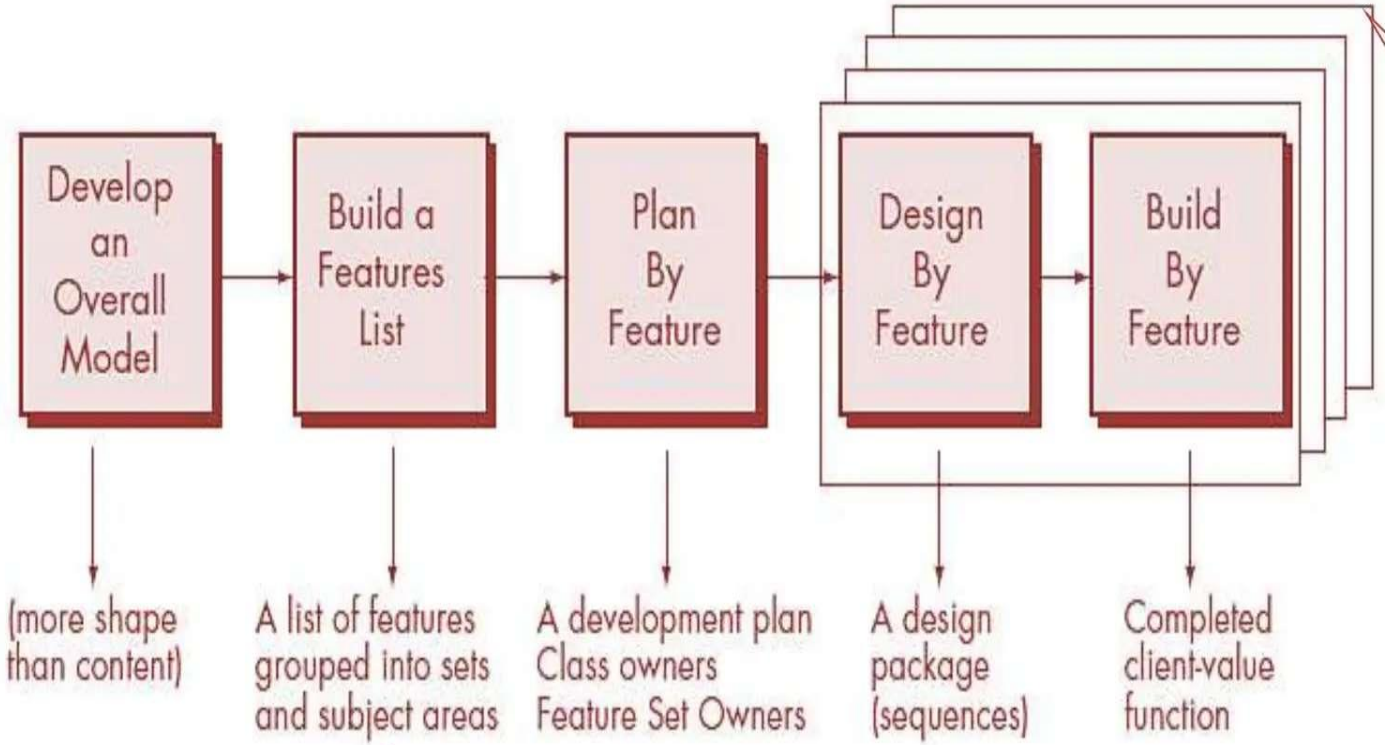
**Postmortems**
Team analyses its own performance and makes appropriate improvements.

# Dynamic Systems Development Methods (DSDM)

Various phases of this life cycle model

- **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

# Feature Driven Development (FDD)




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

In FDD, the feature means client valued function.

# Feature Driven Development (FDD) cont.

## 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