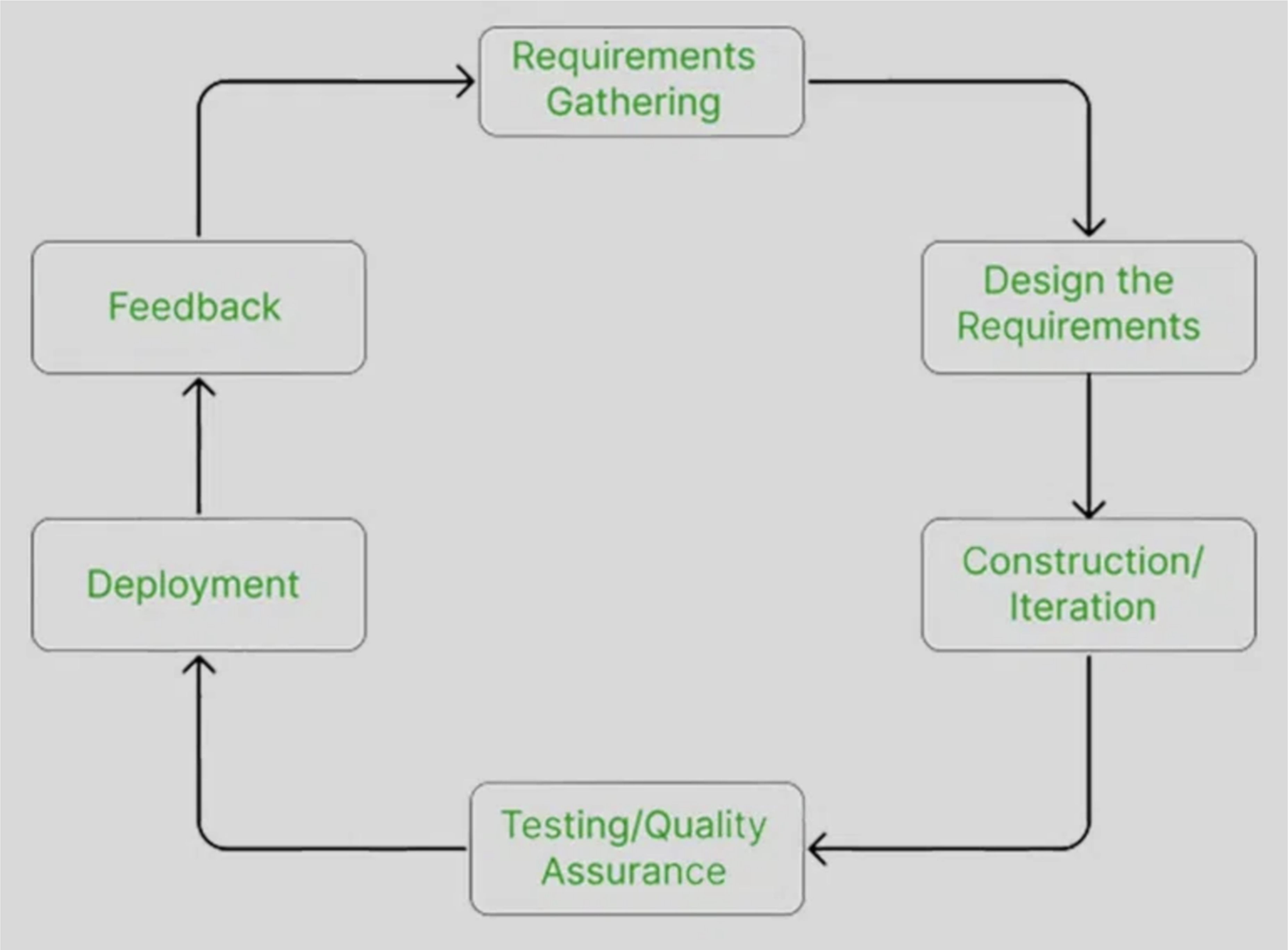
Unit - 4

Agile Model

"Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.


Requirements Gathering
Design the Requirements
Construction/ Iteration
Testing/Quality Assurance
Deployment
Feedback

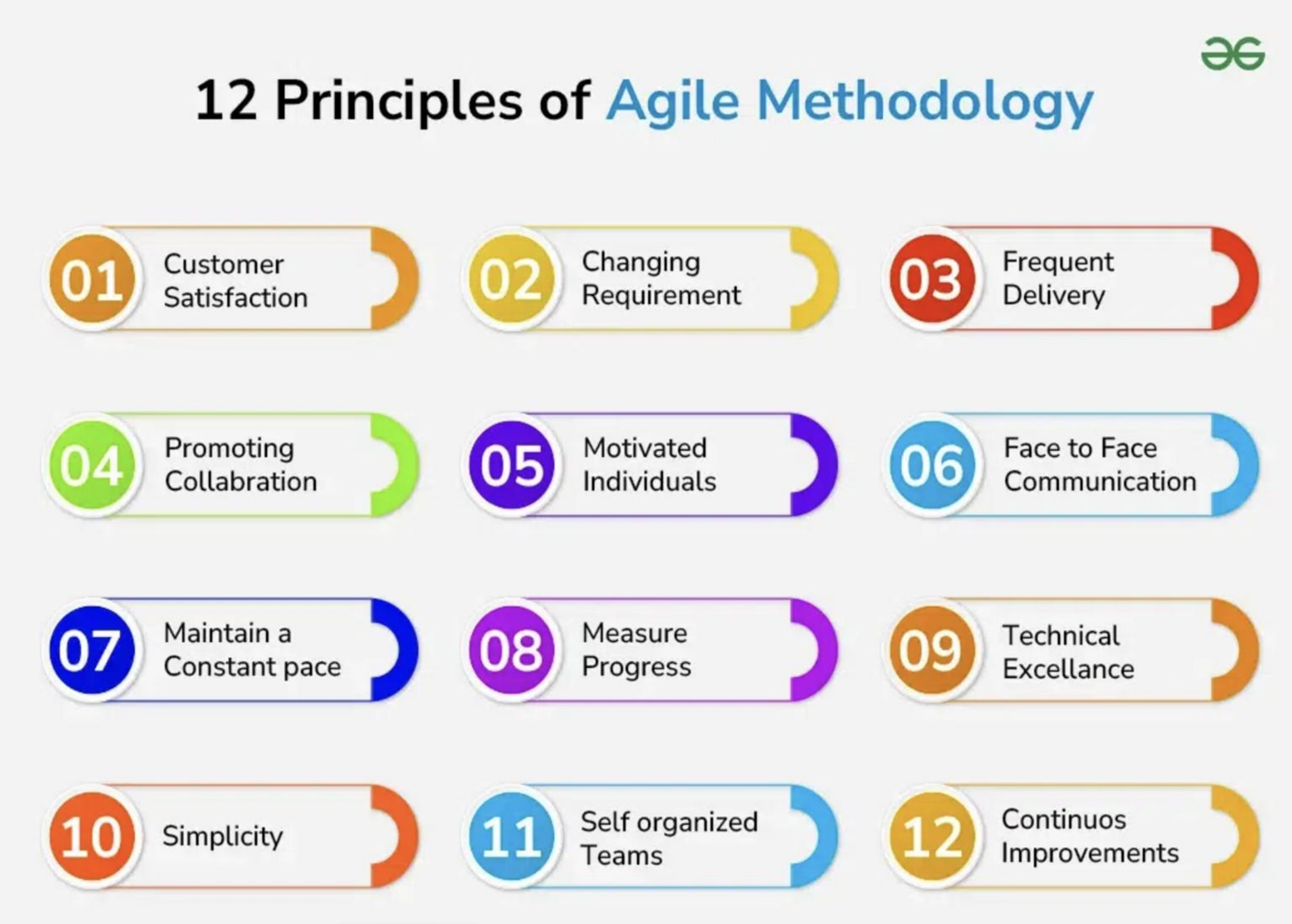
1. **Requirement Gathering:-** In this step, the development team must gather the requirements, by interaction with the customer. Development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.
2. **Design the Requirements:-** In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wireframing and designing user interfaces are done in this phase.
3. **Construction / Iteration:-** In this step, development team members start working on their project, which aims to deploy a working product.

4. Testing / Quality Assurance:- Testing involves Unit Testing, Integration Testing, and System Testing. A brief introduction of these three tests is as follows:

- **Unit Testing:-** Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.
- **Integration Testing:-** Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.
- **System Testing:-** Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5. Deployment:- In this step, the development team will deploy the working project to end users.

6. Feedback:- This is the last step of the **Agile Model.** In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.



12 Principles of Agile Methodology

01 Customer Satisfaction

02 Changing Requirement

03 Frequent Delivery

04 Promoting Collabration

05 Motivated Individuals

06 Face to Face Communication

07 Maintain a Constant pace

08 Measure Progress

09 Technical Excellance

10 Simplicity

11 Self organized Teams

12 Continuos Improvements

1. Our highest priority is to satisfy the client through early and continuous delivery of valuable computer software.
2. Welcome dynamic necessities, even late in development. Agile processes harness modification for the customer's competitive advantage.
3. Deliver operating computer software often, from a pair of weeks to a couple of months, with a preference to the shorter timescale.
4. Business individuals and developers should work along daily throughout the project.
5. The build comes around actuated people. offer them the setting and support they have, and trust them to urge the task done.

6. the foremost economical and effective methodology of communication info to and among a development team is face-to-face speech.

7. Working with computer software is the primary life of progress.

8. Agile processes promote property development. The sponsors, developers, and users will be able to maintain a relentless pace indefinitely.

9. Continuous attention to technical excellence and smart style enhances nimbleness.

10. Simplicity—the art of maximizing the number of work not done—is essential.

11. The most effective architectures, necessities, and styles emerge from self–organizing groups.

12. At regular intervals, the team reflects on a way to become simpler, then tunes and adjusts its behavior consequently.

When To Use the Agile Model?

- When frequent modifications need to be made, this method is implemented.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with the team all the time.
- when the project needs to be delivered quickly.
- Projects with few regulatory requirements or not certain requirements.
- projects utilizing a less-than-strict current methodology
- Those undertakings where the product proprietor is easily reachable
- Flexible project schedules and budgets.

Advantages of Agile Model

- **Flexibility**: Easily adapts to changing requirements.
- **Customer Collaboration**: Continuous involvement ensures alignment with needs.
- **Early Delivery**: Delivers usable parts of the product early.
- **Improved Quality**: Frequent testing and integration.
- **Team Empowerment**: Encourages accountability and collaboration.
- **Risk Mitigation**: Identifies and addresses risks early.
- **Transparency**: Keeps stakeholders informed and involved.

Disadvantages of Agile Model

- **Unpredictable Timeline and Budget**: Hard to estimate due to changing scope.
- **High Customer Involvement**: Requires regular feedback, which may be demanding.
- **Resource Intensive**: Continuous testing and iterations need resources.
- **Requires Skilled Team**: Agile benefits greatly from experienced team members.
- **Less Documentation**
- **Maintenance Problem**

Agile Frameworks/Methodologies:

- eXtreme Programming(XP)
- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development

Extreme Programming (XP)

Extreme Programming (XP) is an agile software development methodology. Extreme Programming (XP) is a way of creating software that focuses on making things simple, flexible, and high-quality by working closely with customers and the team. Imagine building a project step-by-step, checking in often with the people who will use it to make sure it’s exactly what they need.

Here’s how it works in simple terms:

1. **Teamwork**: Two programmers work together on one computer. One writes the code while the other reviews it. This way, mistakes get caught early, and both people learn from each other.
2. **Testing First**: Before writing any actual code, developers write tests to check if the code works as expected. This makes sure that everything they build is correct and reliable from the start.
3. **Small Steps**: Instead of waiting to test or combine code at the end, developers combine their code with the main project many times a day. This helps catch issues right away, so nothing big goes wrong later.

1. **Customer Involvement**: A customer or user is involved throughout the project, answering questions, and giving feedback. This way, the team knows they’re building something that fits what the customer really wants.
2. **Simple Design**: XP focuses on keeping things as simple as possible. The team builds only what’s needed right now, so they can easily add or change things if the customer’s needs evolve.
3. **Frequent Updates**: Instead of delivering a big project all at once, the team shares small updates often. The customer can see progress, give feedback, and ensure the project is always on track.

XP Values

- Communication: Face to Face communication with client
- Simplicity: Design should be simple
- Feedback: Take feedback from end user and customer
- Courage: Courage to change and re-code
- Respect: Respect your team member

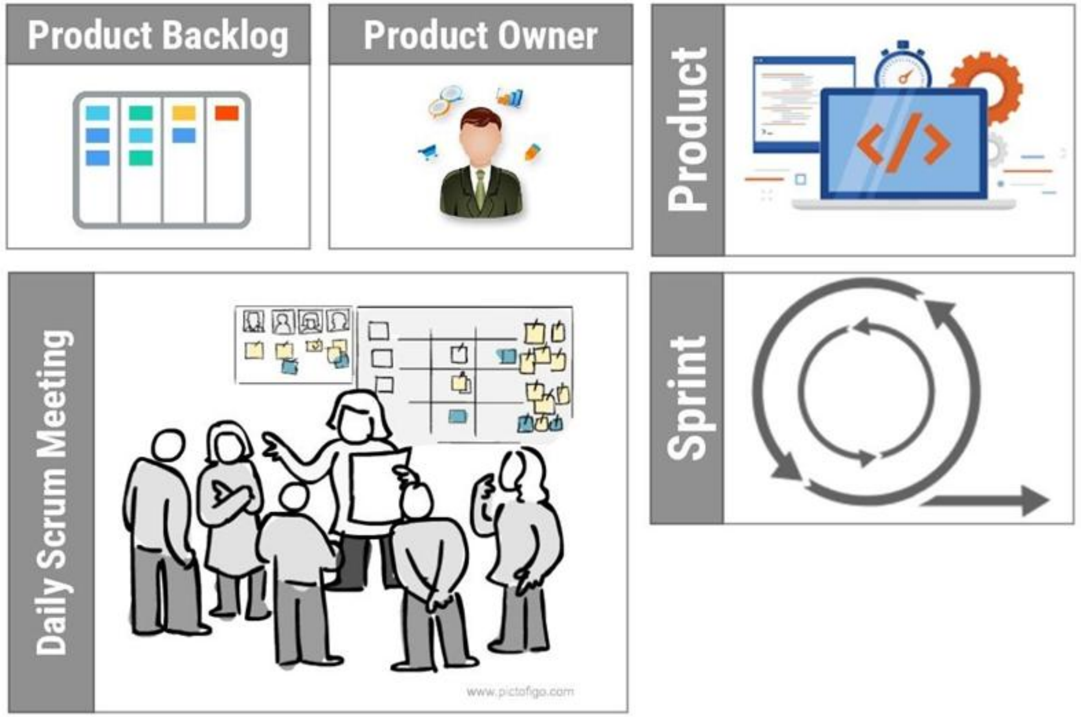
What is Scrum?

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

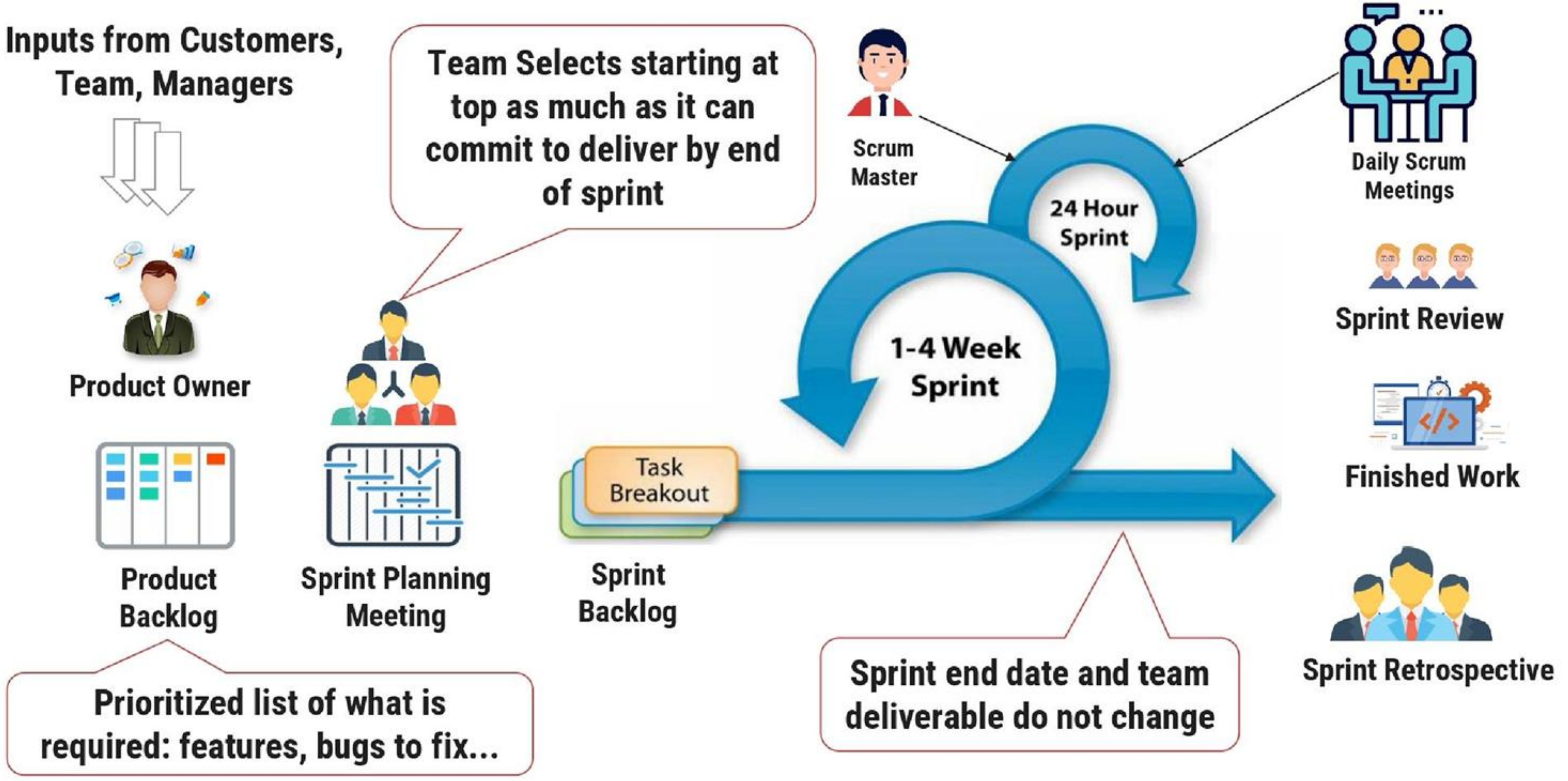
Product

Daily Scrum Meeting

Sprint

www.pictofigo.com

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

Sprint Review

Finished Work

Task Breakout

Product Backlog

Sprint Planning Meeting

Sprint Backlog

Sprint Retrospective

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

Sprint end date and team deliverable do not change

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**

24 Hour
Sprint
1-4 Week
Sprint
Task
Breakout

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

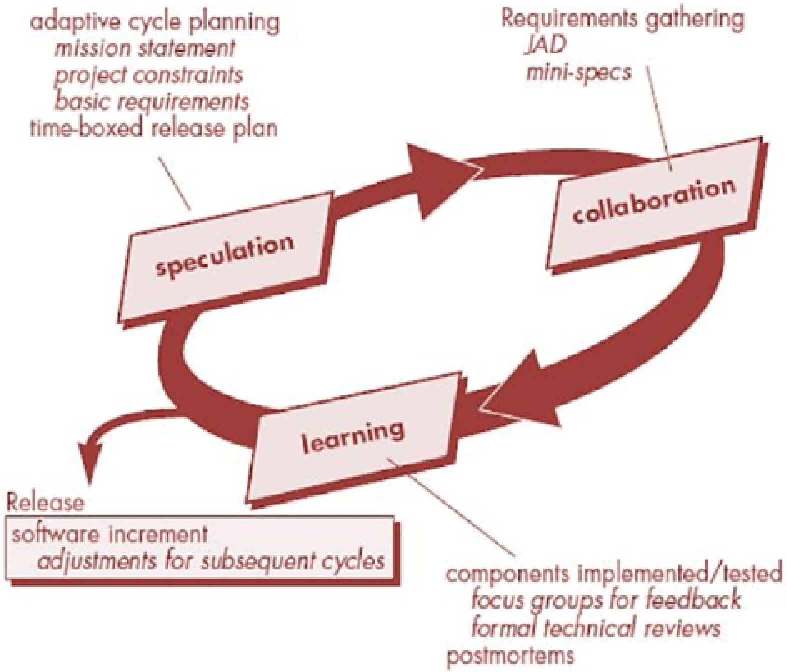
- Deliver **software increment** to customer
- Implemented functionalities are **demonstrated** to the customer

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.

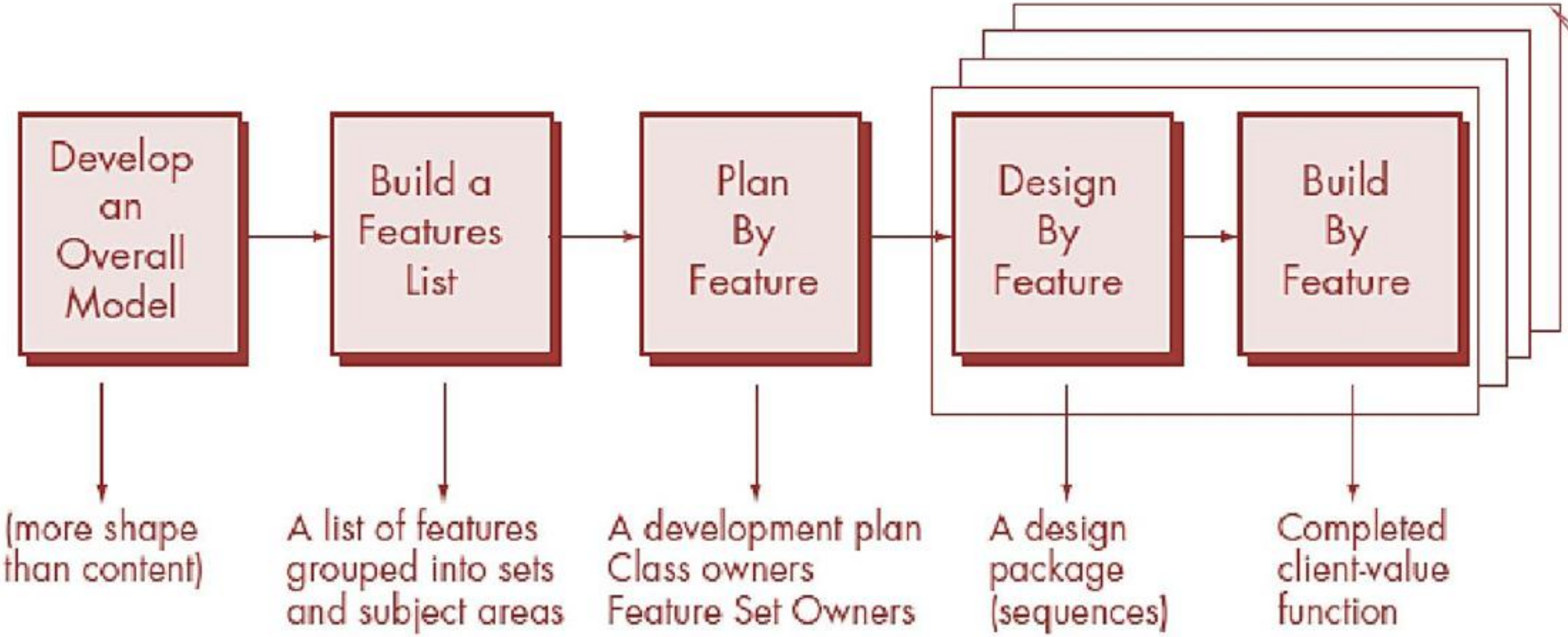
Postmortems
Team analyses its own performance and makes appropriate improvements.

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)



Develop an Overall Model → Build a Features List → Plan By Feature → Design By Feature → Build By Feature

Develop an Overall Model: (more shape than content)

Build a Features List: A list of features grouped into sets and subject areas

Plan By Feature: A development plan
Class owners
Feature Set Owners

Design By Feature: A design package (sequences)

Build By Feature: 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) 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

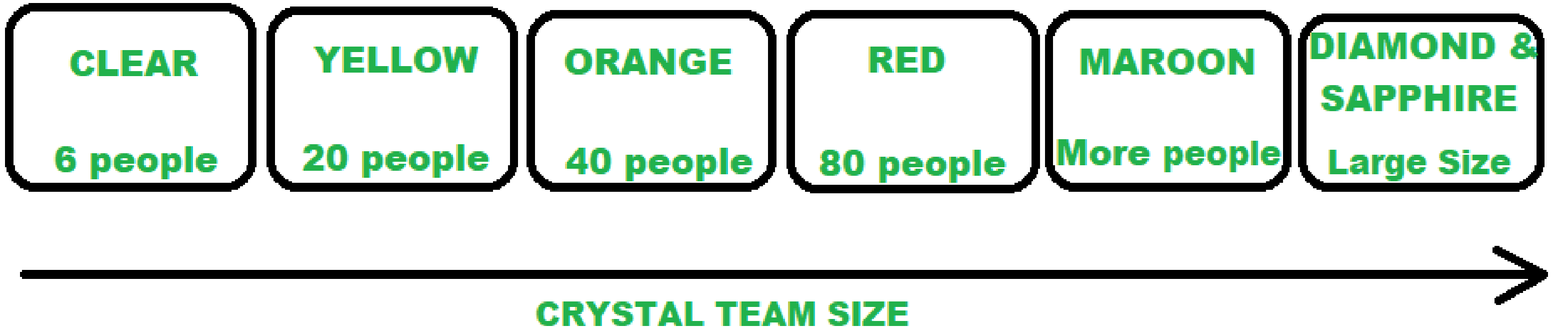
- For each feature the sequence diagram is created

Build by feature

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

Crystal

- The crystal method is an agile framework that focuses on individuals and their interactions.
- The methods are color-coded to significant risk to human life.
- It is mainly for short-term projects by a team of developers working out of a single workspace.
- Crystal family consists of many variants like
  1. Crystal Clear
  2. Crystal Yellow
  3. Crystal Red
  4. Crystal Sapphire
  5. Crystal Red
  6. Crystal Orange Web
  7. Crystal Diamond



Agile Modelling (AM)

- It is a practice-based methodology for effective modeling and documentation of software-based systems.
- It is a collection of values, principles and practices for modeling software that can be applied on a software development project in an effective and light-weight manner.
- Unique principles of AM are :
  - **Use multiple models**
  - **Travel Light**
  - **Content is more important than representation**
  - **Know the models and the tools you use to create them**
  - **Adapt locally**