
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# SWE 2029

# Agile Development Process

# Module - 1

*Dr. Rajesh M*
*School of Computer Science and Engineering*
*Vellore Institute of Technology [VIT]*
*Chennai, India.*

| Module | Topics | L Hrs |
|---|---|---|
| 1 | **INTRODUCTION TO AGILE :**<br>Introduction to Agile Software Process Model - Agile Methodology & Principles – Types – Benefits - Life Cycle, Agile Project Management – Design and Construction - Agile Testing - Agile Tools. | 6 |

# Introduction to

# Agile Software Process Model

# Introduction

- **Process:**

  Splitting of development work into distinct phases / stages;

- **Process Model / Methodology:**

  Contains specific deliverables and artifacts;

- **Example:**

  Waterfall, Spiral, Prototyping, Agile.

# Traditional vs Agile

- Traditional models are based on "**Predictive approach**";
- Agile model is based on "**Adaptive approach**";

**Predictive** – model usually have detailed planning, have complete forecast;

- Entirely depend on requirement analysis and planning done;

**Adaptive** - no detailed planning

- Have clarity on what features need to be developed.

- **Customer interaction** is the backbone of Agile methodology, and open communication with minimum documentation.

# What Does It Mean to be Agile?

**Agile Methods – Goal:** Organization to be agile.

What it means?

Being Agile means:

***Deliver quickly, Change quickly, Change often;***

- Agile methods are many; but all share common characteristics; **Iterative development;**
- Focus on **interaction, communication**, and the reduction of intermediate artifacts;
- Being Agile involves more than simply following guidelines that make a project Agile.
- True agility is more than a collection of practices; it's a frame of mind.

# Software Process Models

# Software Process Models

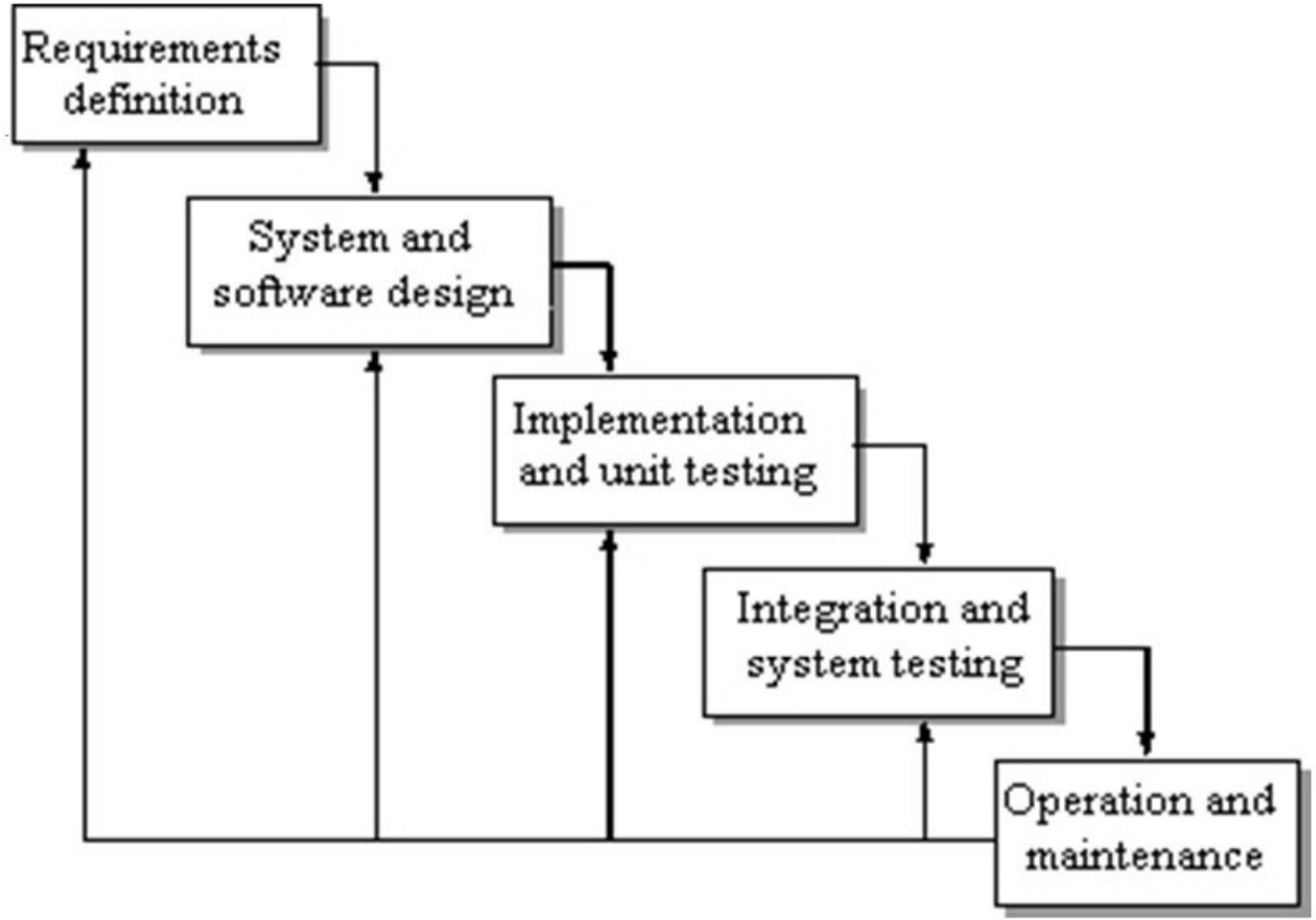
- Waterfall Model
- Iterative Model
- Spiral Model
- V Model
- Prototype
- Agile

# Waterfall Model

# Waterfall Model

- Requirements are very *well documented, clear* and *fixed*.
- Product definition is *stable*.
- Technology is *understood* and is *not dynamic*.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

# Waterfall Model

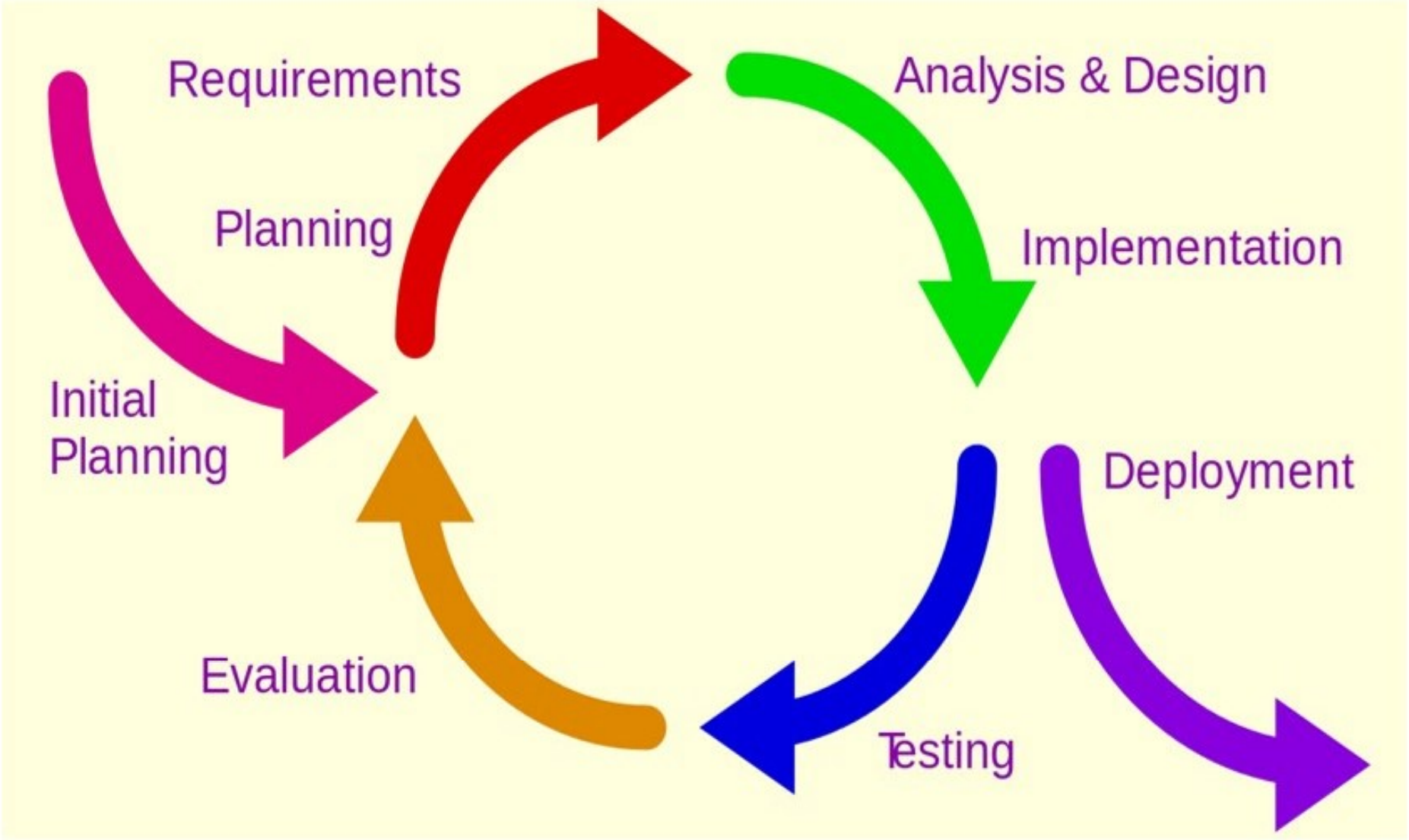


# Iterative Model

# Iterative Model

- Requirements are *clearly defined* and *understood*.
- Major requirements must be defined; some functionalities may evolve with time.
- A *new technology* is being used and is *being learnt by the development team while working on the project*.
- *Resources* with needed skill set are *not available* and are *planned to be used on contract basis* for specific iterations.
- There are some *high risk features and goals* which may change in the future.

# Iterative Model

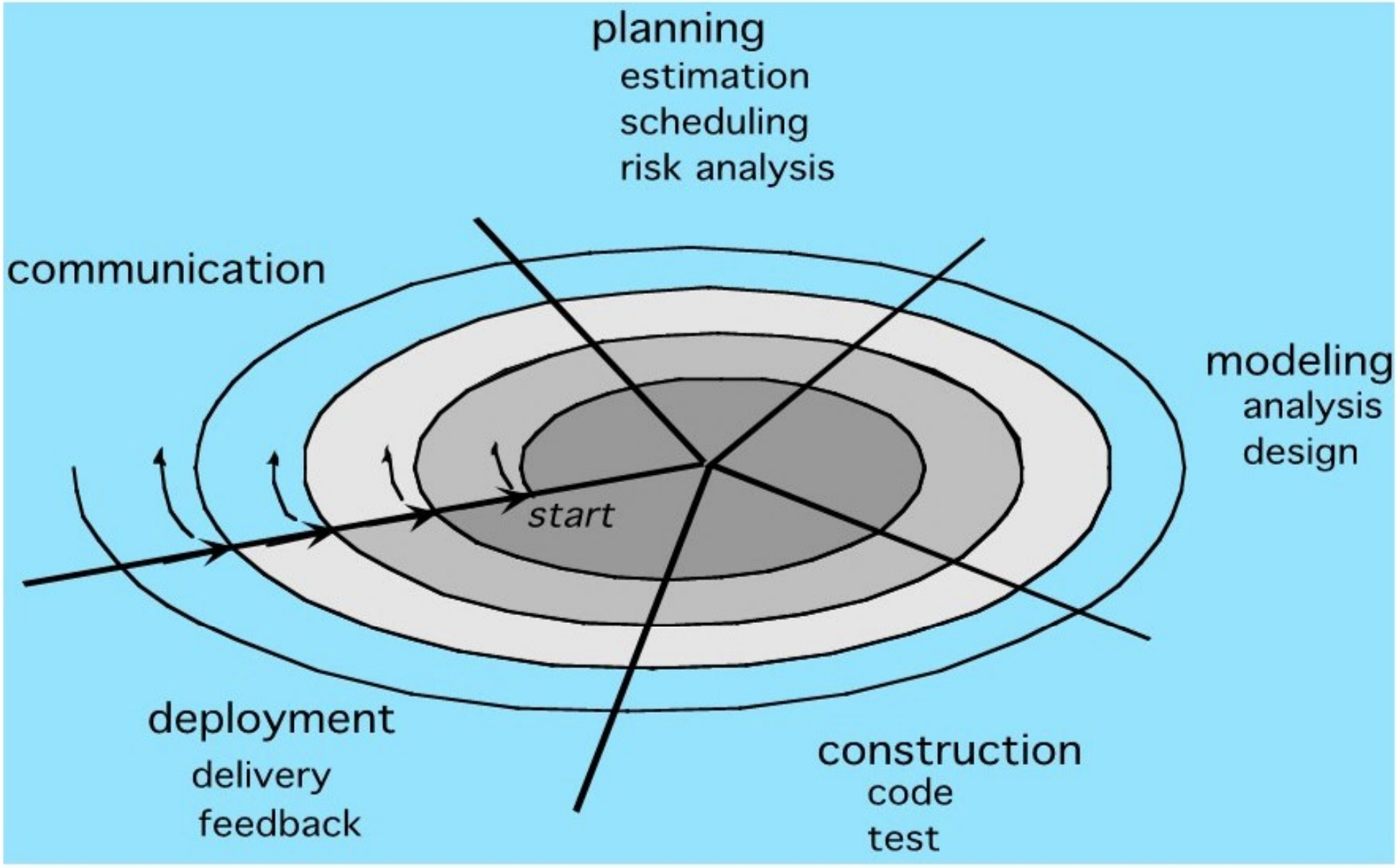


# Spiral Model

# Spiral Model

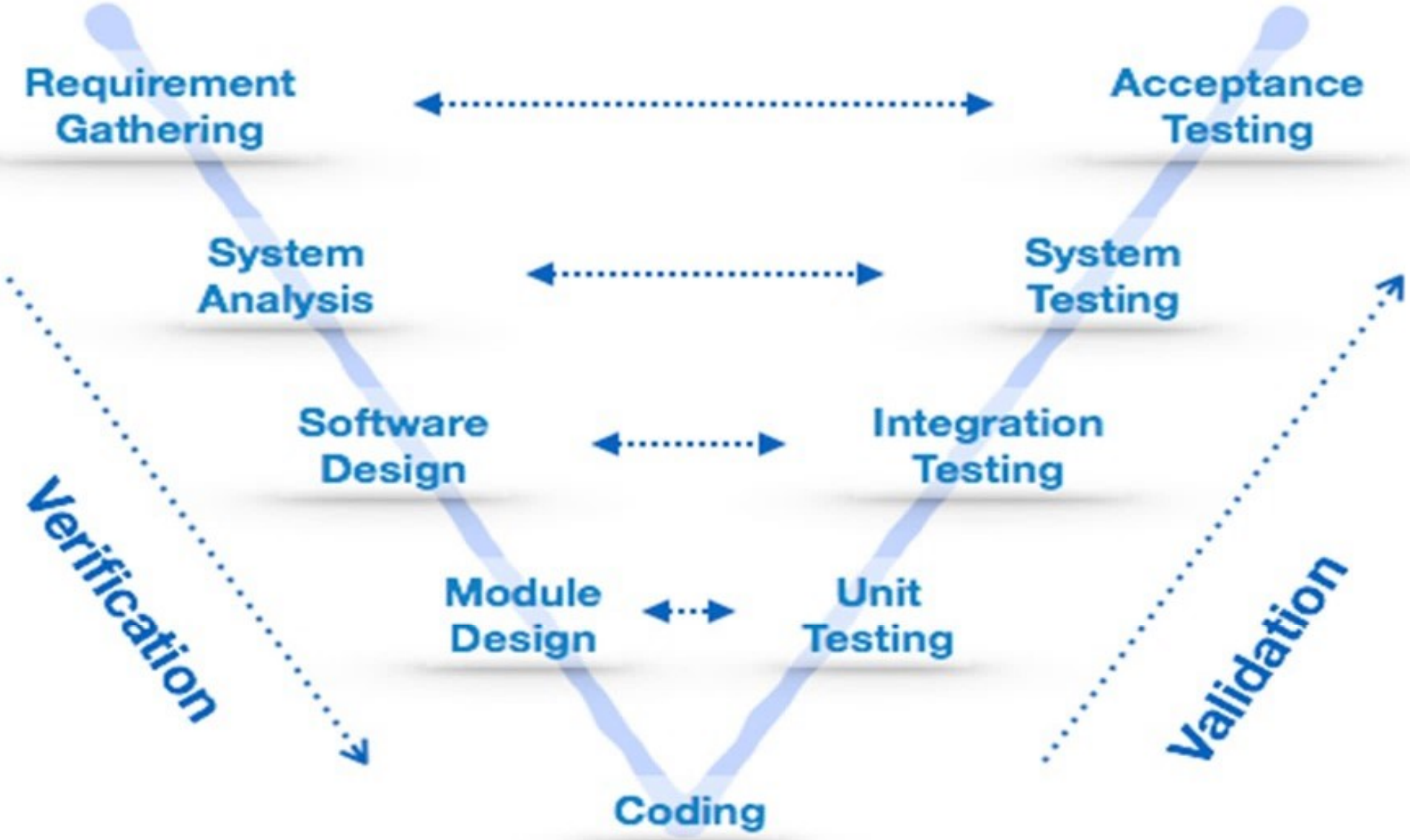
- *Process is represented as a spiral* rather than as a sequence of activities with backtracking.
- *Each loop in the spiral* represents a *phase in the process*.
- When there is a *budget constraint*.
- For *medium to high-risk projects*.
- *Customer is not sure of their requirements* which is usually the case.
- *Requirements are complex* and need evaluation to get clarity.
- New product line which *should be released in phases to get enough customer feedback*.
- *Significant changes are expected* in the product *during the development cycle*.

# Spiral Model - Structure



# V Model

# V Model


Requirement Gathering
Acceptance Testing
System Analysis
System Testing
Software Design
Integration Testing
Module Design
Unit Testing
Coding
Verification
Validation

# V-Model

- Requirements are *well defined, clearly documented* and *fixed*.
- Product definition is *stable*.
- *Technology is not dynamic* and is well understood by the project team.
- There are no ambiguous or undefined requirements.
- The project is *short*.

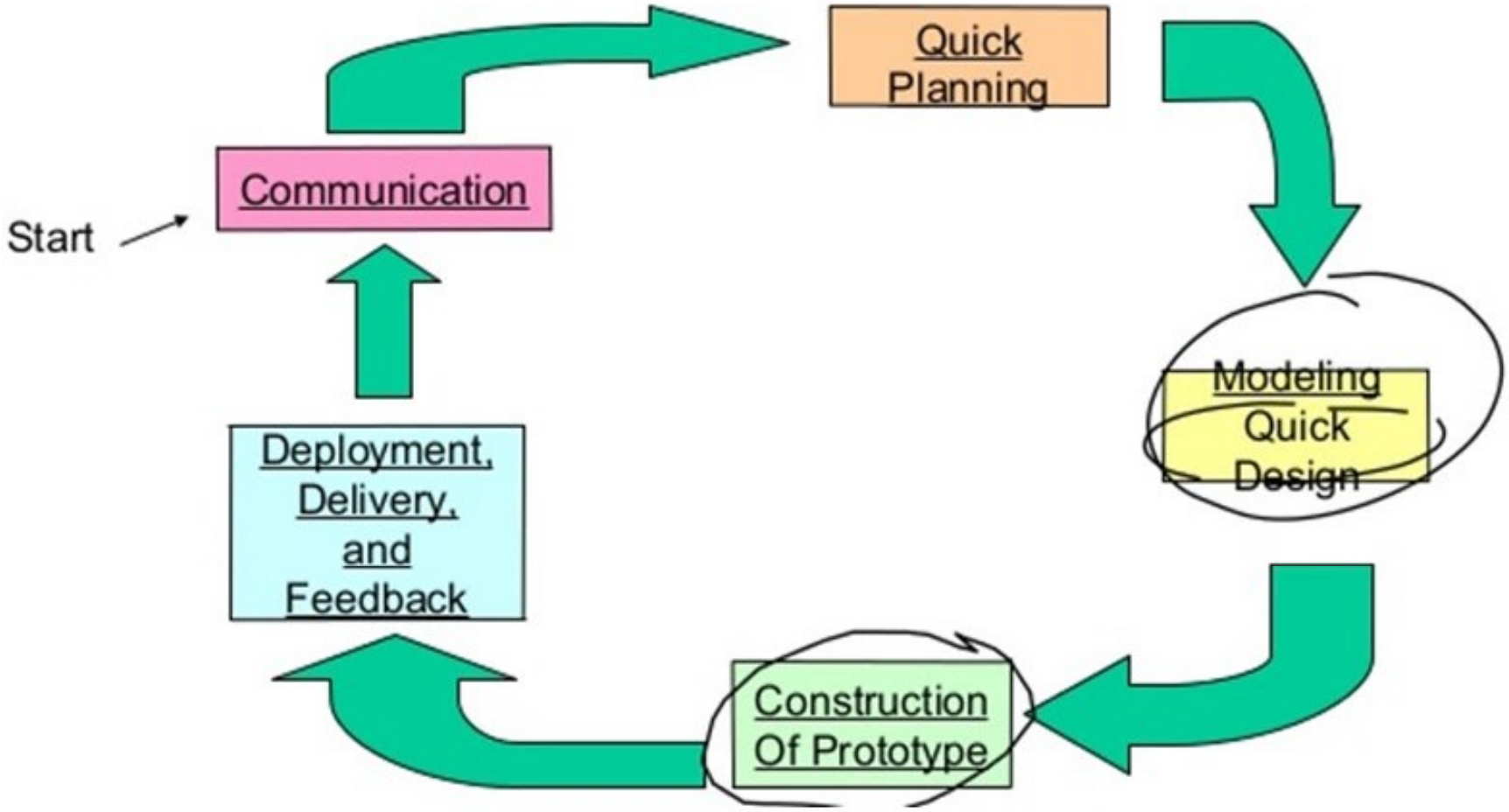
# Prototype

# Software Prototype

Steps involved are:

- Basic *Requirement identification*
- *Developing initial prototype*
- *Review* the prototype
- *Revise* and *enhance*

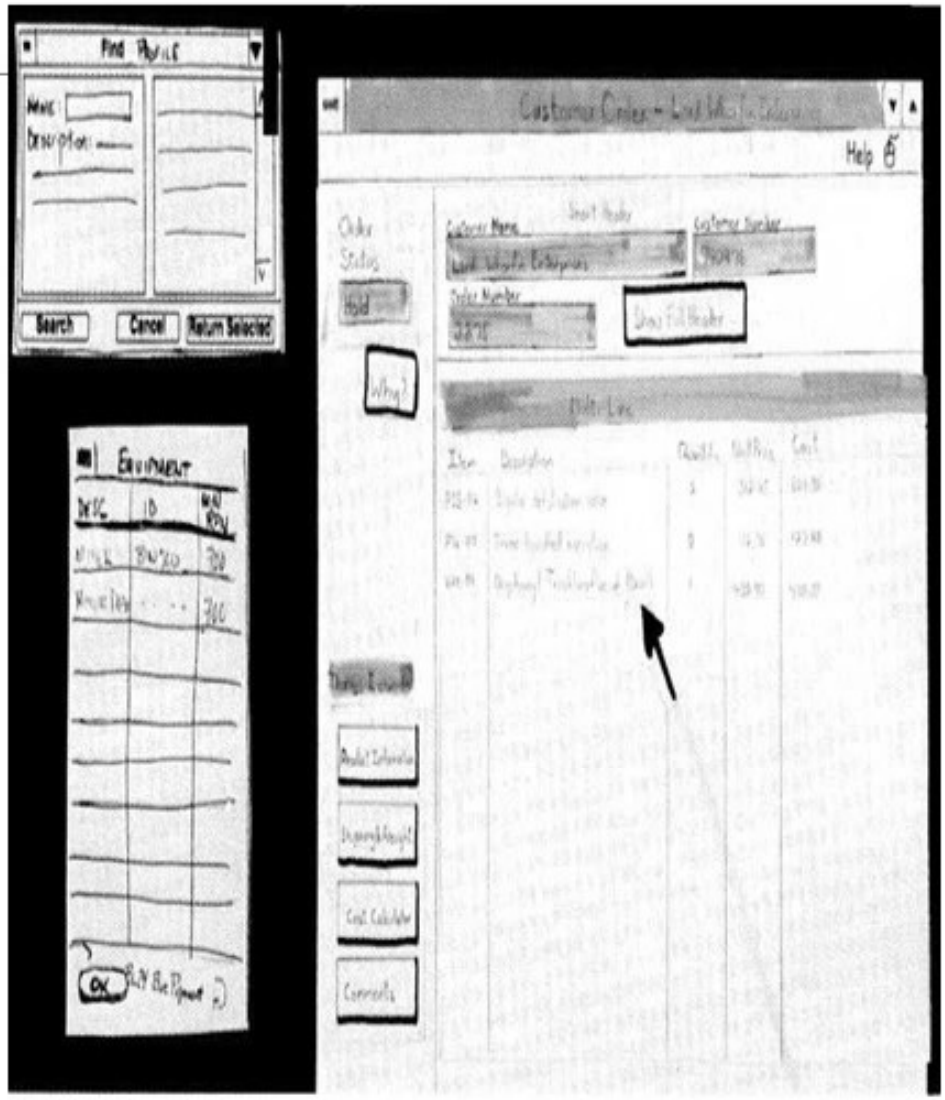
# Software Prototype



# Prototyping

Creating prototypes of software applications i.e. incomplete versions of the software program being developed

A *prototype typically simulates only a few aspects* of, and may be completely different from, the final product.



## Prototype Dimensions:

Horizontal & Vertical dimensions.

## Software Prototyping Types:

- Throwaway/Rapid Prototyping (rapid or close ended prototyping)
- Evolutionary (Breadboard) Incremental & Extreme

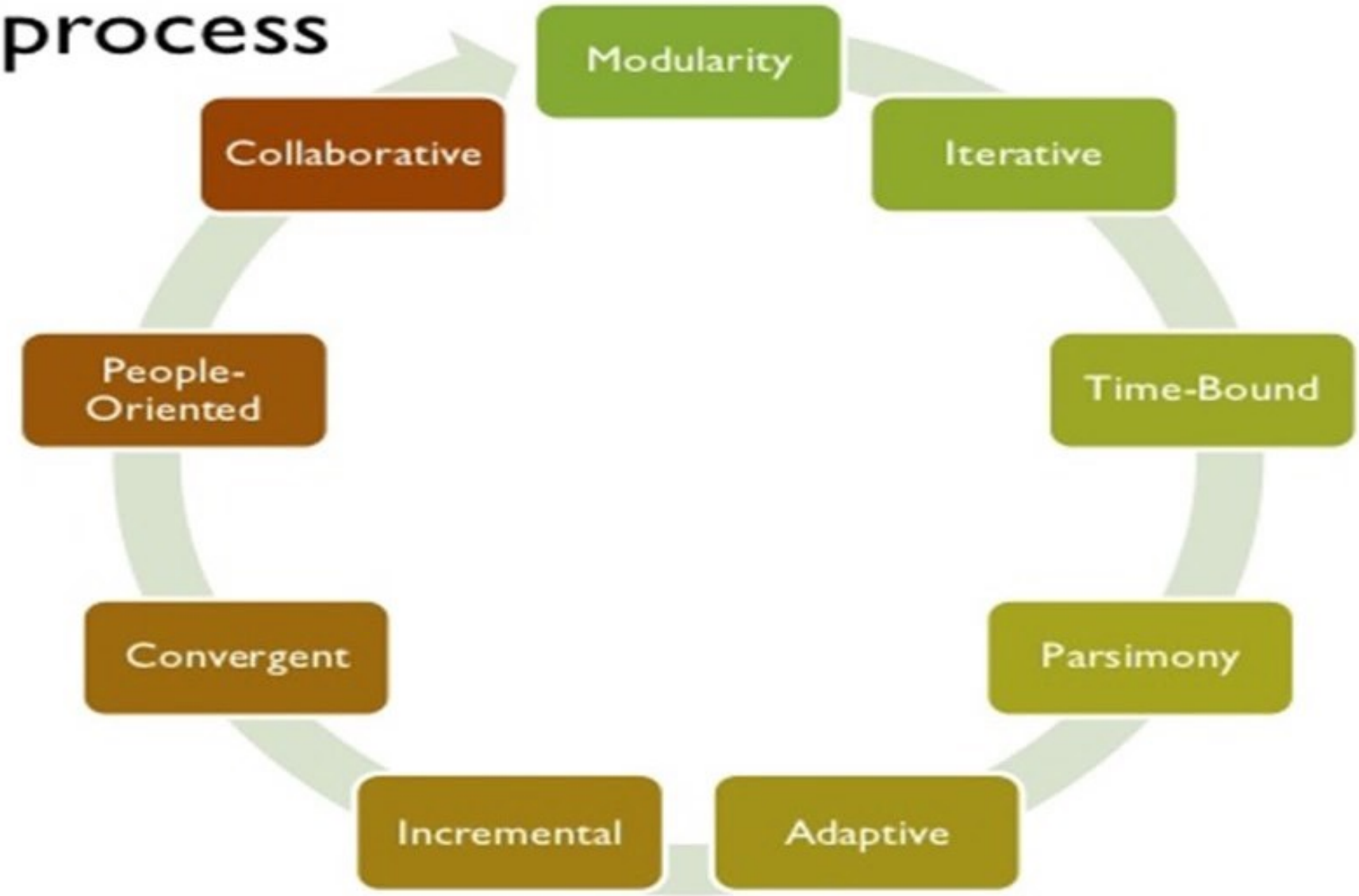
# Agile Process Model

# Agile Process Model

- It is a *combination of iterative and incremental process models.*
- Main focus is on:
  - *process adaptability;*
  - *customer satisfaction,*
  - *rapid delivery of working software product;*

# Agile Process Model

## Agile process



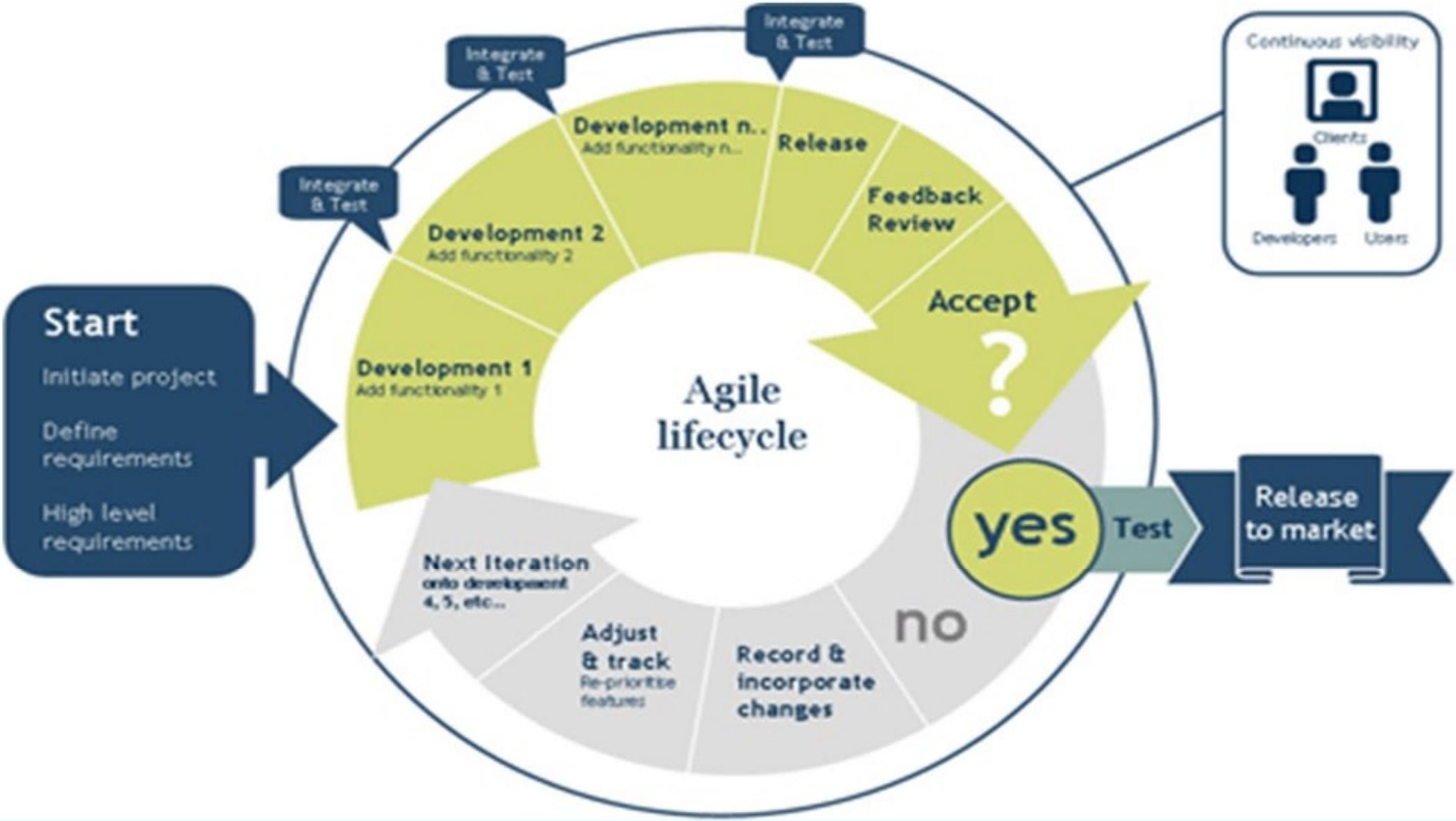
# Agile Process Model

- "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.

# Agile Process Model

- 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.

# Agile Process Life Cycle



# Agile Process Life Cycle

Key Agile Software Development Lifecycle Phases

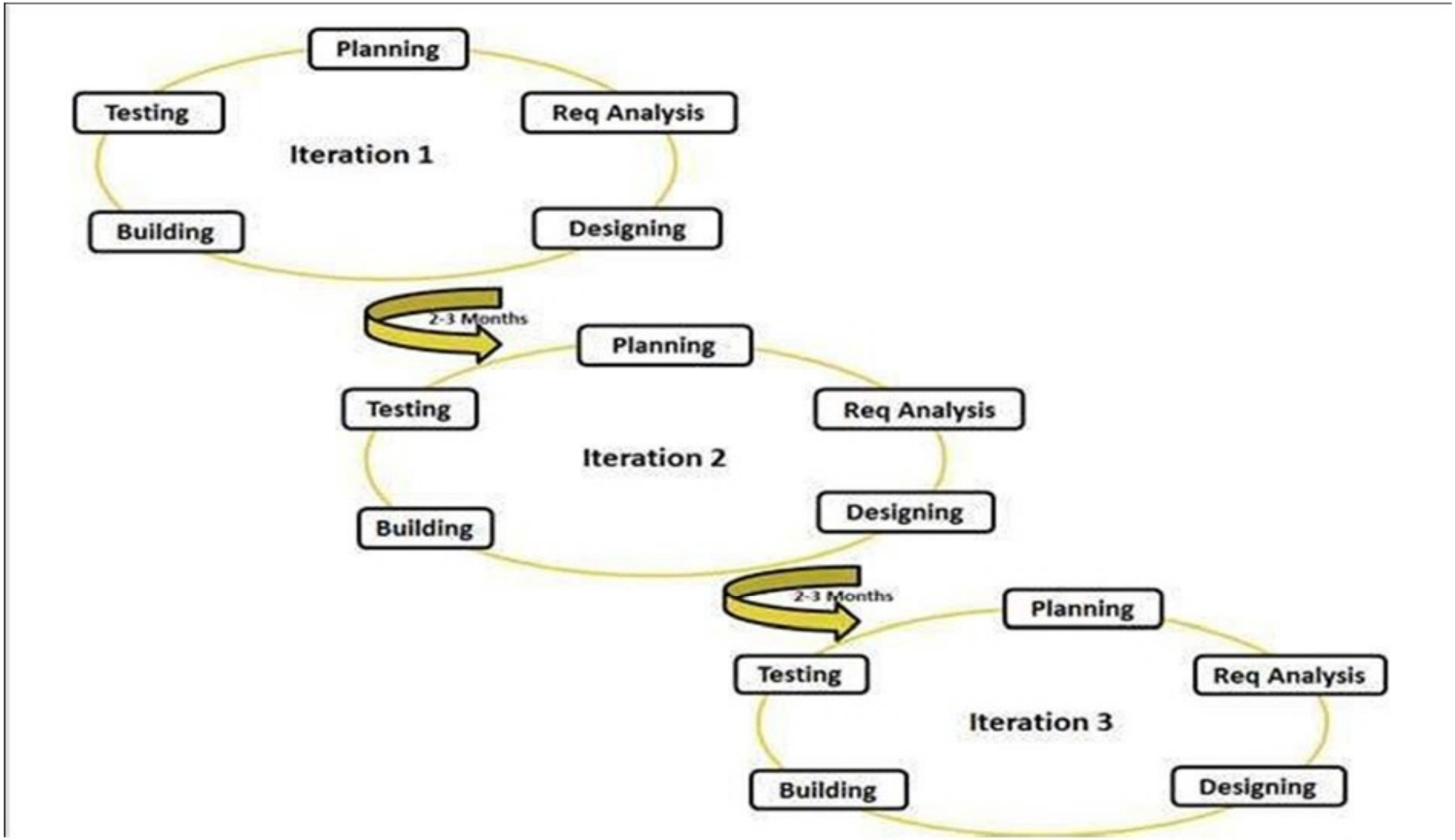
Phase 1: Requirements
Phase 2: Design
Phase 3. Development and Coding
Phase 4. Integration and Testing
Phase 5. Implementation and Deployment
Phase 6. Review

# Agile Process Life Cycle

The Key Principles of Agile Development

- Change is inevitable. The project needs to adapt to it rather than ignore it;
- Delivering results is more important than established processes and tools;
- Real customer's needs take priority over the requirements in the development plans.

# Agile Process Life Cycle



# Agile Methods

**The most popular agile methods include:**

- Scrum (1995),
- Crystal Method (2004),
- Extreme Programming(XP) (1996),
- Feature Driven Development(FDD) (1999)
- Dynamic Systems Development Method (DSD) (1995).
- Agile Unified Process(AUP) (2005) and
- Adaptive Software Development(ASD) (1997),

# Scrum

A **light-weight agile process tool**

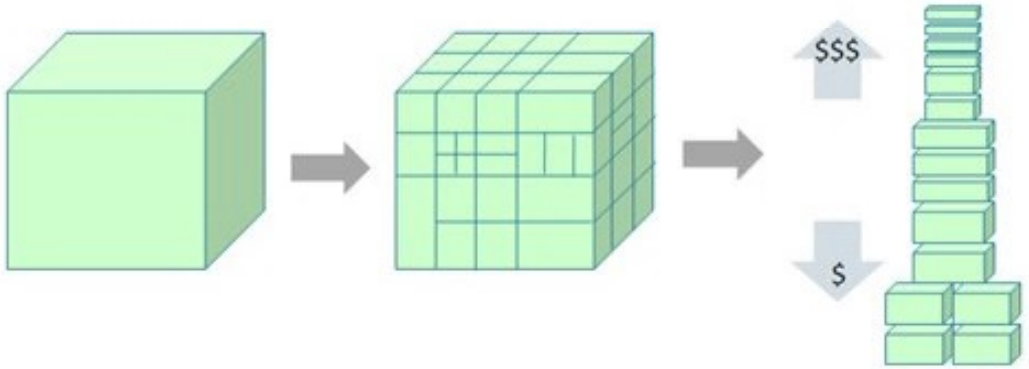
**Split your organization** into small, cross-functional, self-organizing teams.

Product/ Project Owner

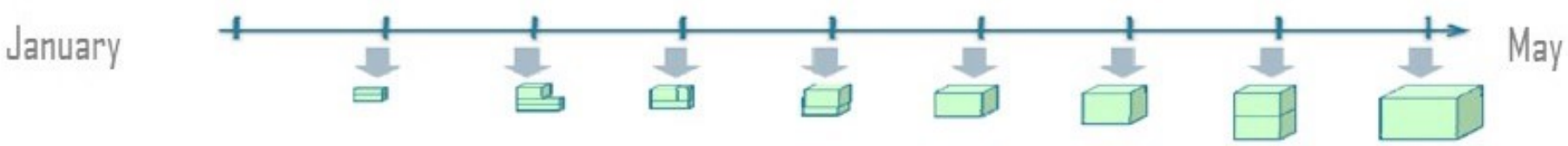
Scrum Master

Scrum Team

**Split your work** into a list of small, concrete deliverables. Sort the list by priority and estimate the relative effort of each item.



# Scrum (contd..)

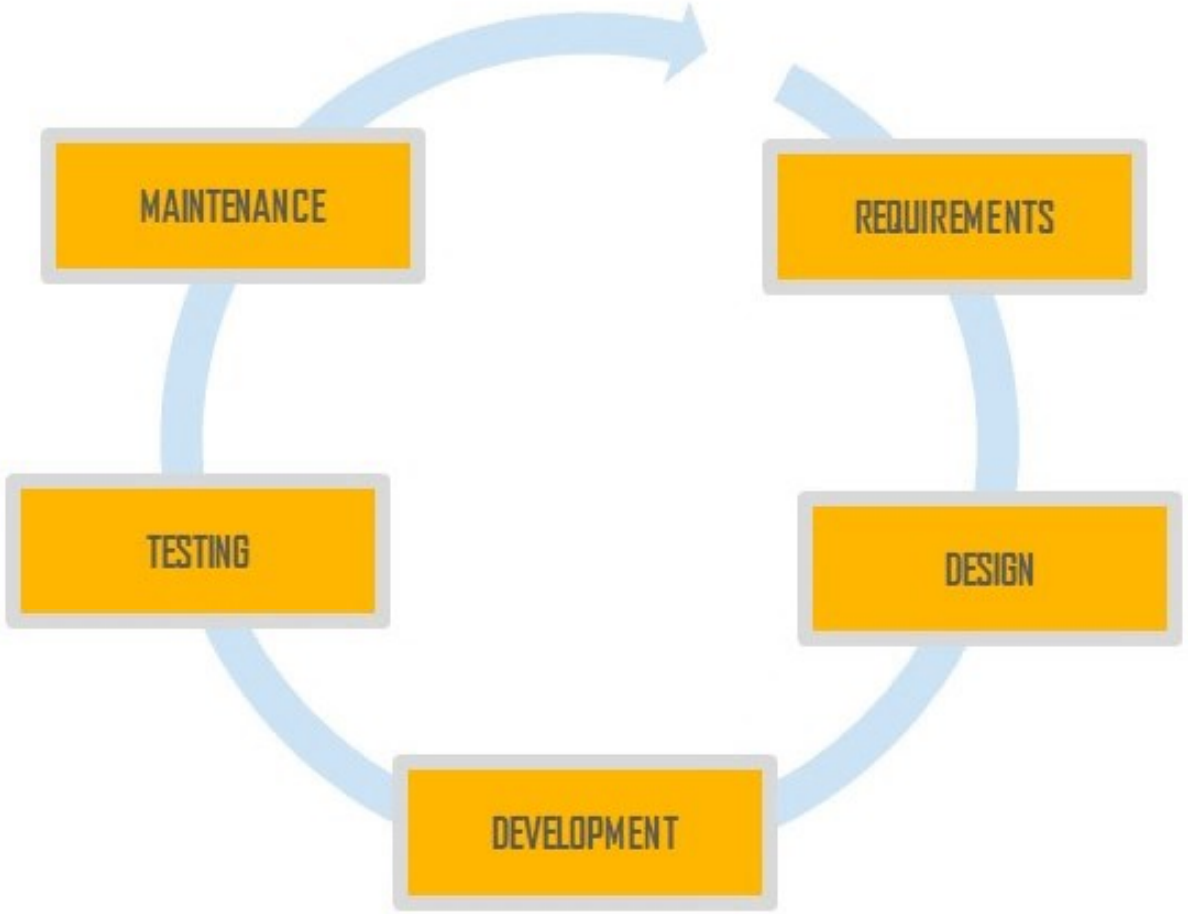



**Split time** into **short fixed-length iterations/ sprints (usually 2 – 4 weeks),** with potentially shippable code demonstrated after each iteration.

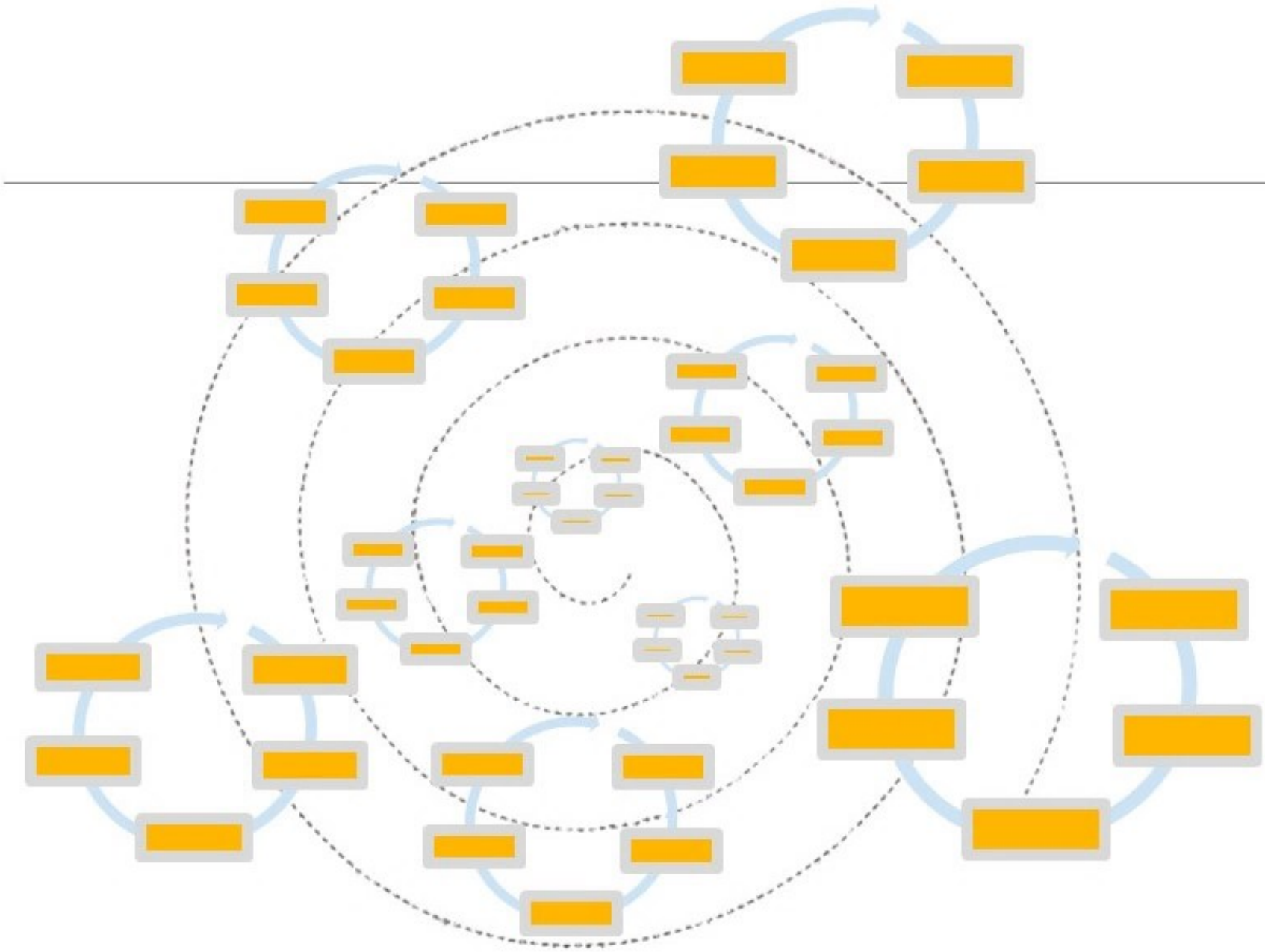
**Optimize the release plan** and **update priorities** in collaboration with the customer, based on insights gained by inspecting the release after each iteration.

**Optimize the process** by having a **retrospective after each iteration**.

# Scrum vs. Waterfall



# Iterative Scrum



# Things we do in Scrum

## Scrum terminologies

The project/ product is described as a **list of features**: the **backlog**.

The **features are described** in terms of **user stories**.

The scrum team **estimates** the **work** associated with each story.

Features in the backlog are **ranked** in order of importance.

**Result:** a **ranked** and **weighted** list of product features, a **roadmap**.

**Daily scrum meeting** to discuss **What did you do y'day? What will you do today? Any obstacles?**

# Crystal Methods

**Crystal Methods:**

- Developed in early 1990s;
- Major obstacles faced by product development was *poor communication* and modelled the Crystal methods to address these needs;
- Replacing written documentation with *face-to- face interactions*, you can improve the likelihood of delivering the system.

# Crystal Methods – contd..

- *"Crystal" focuses on people*, interaction, skills, talents and communication as first order effects on performance. Process remains important, but secondary".

- "Crystal" named to represent a **gemstone**, i.e., each facet is another version of the process, all arranged around an identical core.

# Crystal Methods – contd..

- Different versions in crystal methods are *Crystal Clear, followed by Crystal Yellow, Crystal Orange, Crystal Red,* etc.

- **Clear** - for teams of 8 or fewer people
- **Yellow** - for teams of 10-20 people
- **Orange** - for teams of 20-50 people
- **Red** - for teams of 50-100 people

# Crystal Methods – contd..

- Crystal method is an agile software development approach that focuses primarily on people and their interactions when working on a project rather than on processes and tools. Alistair believed that **people's skills and talents** as well as the way **they communicate** has the **biggest impact on the outcome of the project.**

Crystal Method is based on **two fundamental assumptions**:

- **Teams can streamline their processes** as their work and become a more optimized team
- **Projects are unique and dynamic** and require specific methods

# Extreme Programming (XP)

# Extreme Programming

- Most prominent Agile Software development method.
- Prescribes a set of daily stakeholder practices.
- "Extreme" levels of practicing leads to more responsive software.
- Changes are more realistic, natural, inescapable.

# XP

- ***Open workspace***: **Developers work in a common workspace set up** with individual workstations and **common development machines in the center.**

# Extreme Programming Project

# XP: Why?

**Previously:**

- **Get all the requirements** before starting design
- **Freeze the requirements before starting** development
- **Resist changes:** they will **lengthen schedule**
- **Build a change control process** to ensure that proposed changes are looked at carefully and no change is made without intense scrutiny
- **Deliver a product that is obsolete** on release

# XP: Embrace Change

- Recognize that:
  - **All requirements will not be known at the beginning**
  - Requirements will change

- **Use tools to accommodate change** as a natural process
- **Do the simplest thing** that could possibly work and refactor
- Emphasize values and principles rather than process

# Feature Driven Development

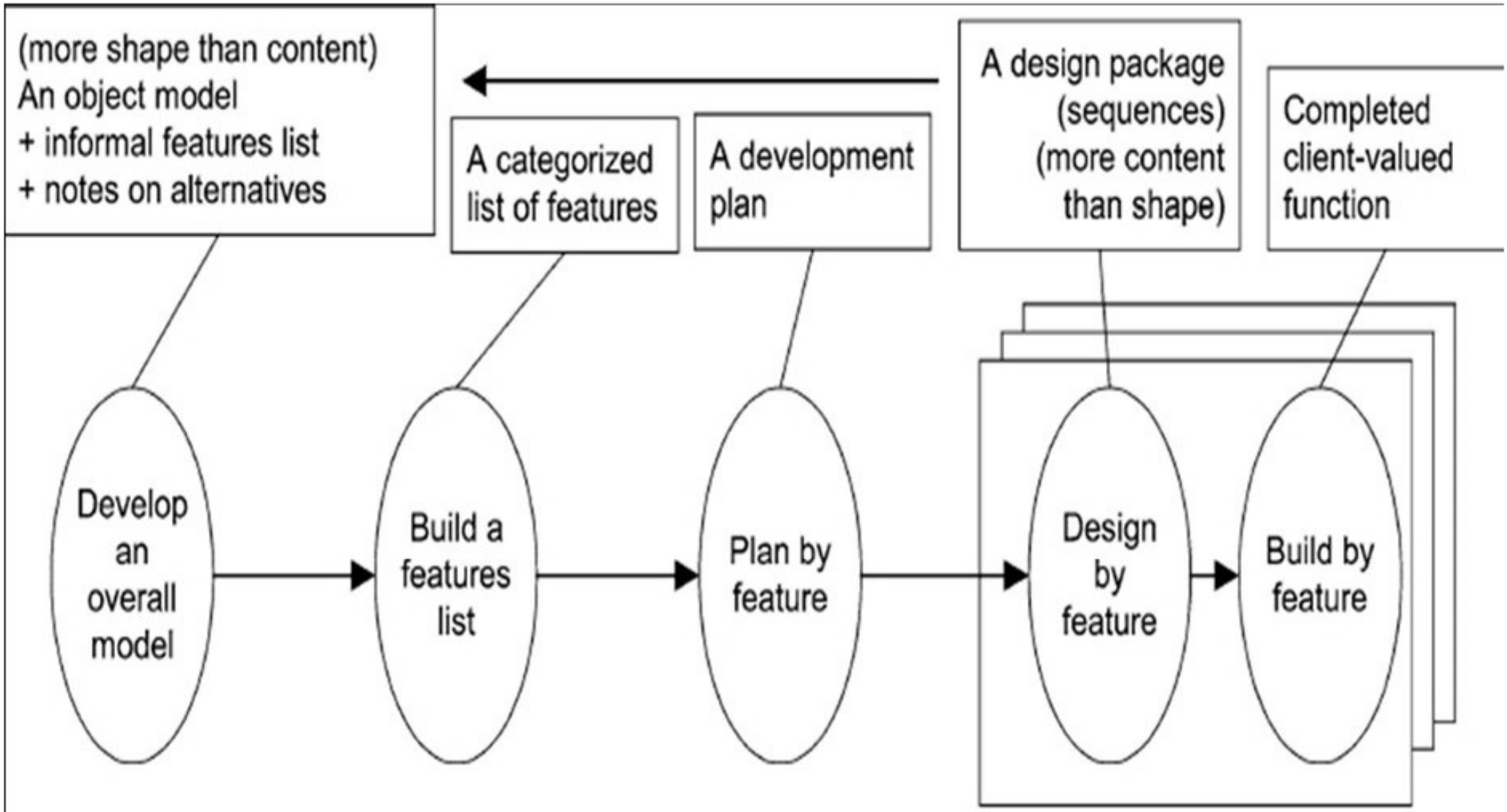
# Feature Driven Development

Introduced during late 1990s.

Core values of FDD are:

- A model for building any system is necessary in order to scale to larger projects.
- A simple, well-defined process works best. Process steps should be logical.
- Good processes move to the background so the team members can focus on results.
- **Short, iterative, feature-driven life cycles are best.**

# FDD Process


(more shape than content)
An object model
+ informal features list
+ notes on alternatives
A categorized list of features
A development plan
A design package (sequences) (more content than shape)
Completed client-valued function
Develop an overall model
Build a features list
Plan by feature
Design by feature
Build by feature

# Agile Methods:

## Dynamic Systems Development(DSD) Method

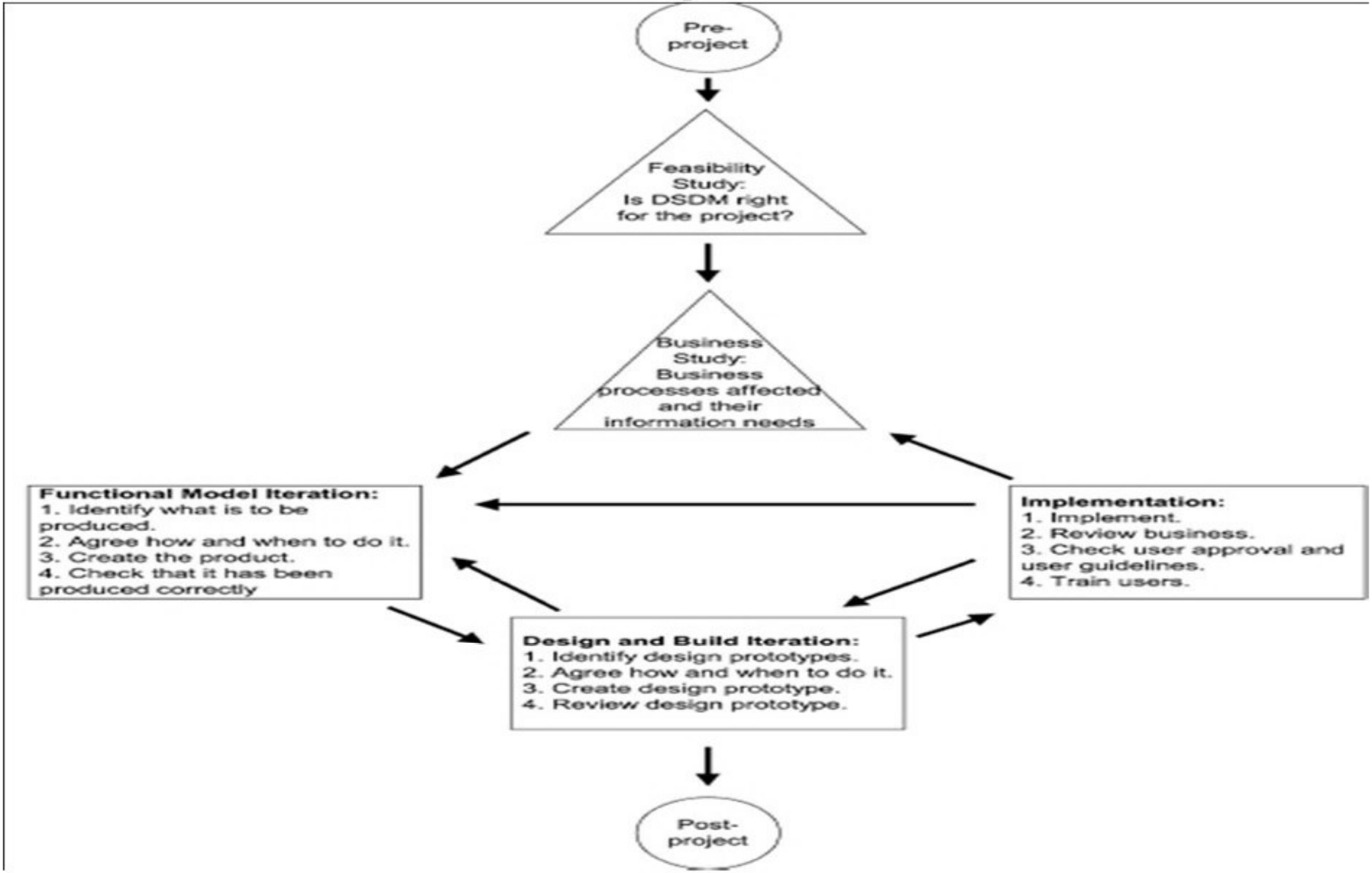
# Dynamic Systems Development Method(DSD)

- Arising in **early 1990's**, it is a formalization of RAD practices;

**DSDM lifecycle has six stages:**

- **Pre-project, Feasibility Study, Business Study, Functional Model Iteration, Design and Build Iteration, Implementation and Post-project.**

# DSD Lifecycle



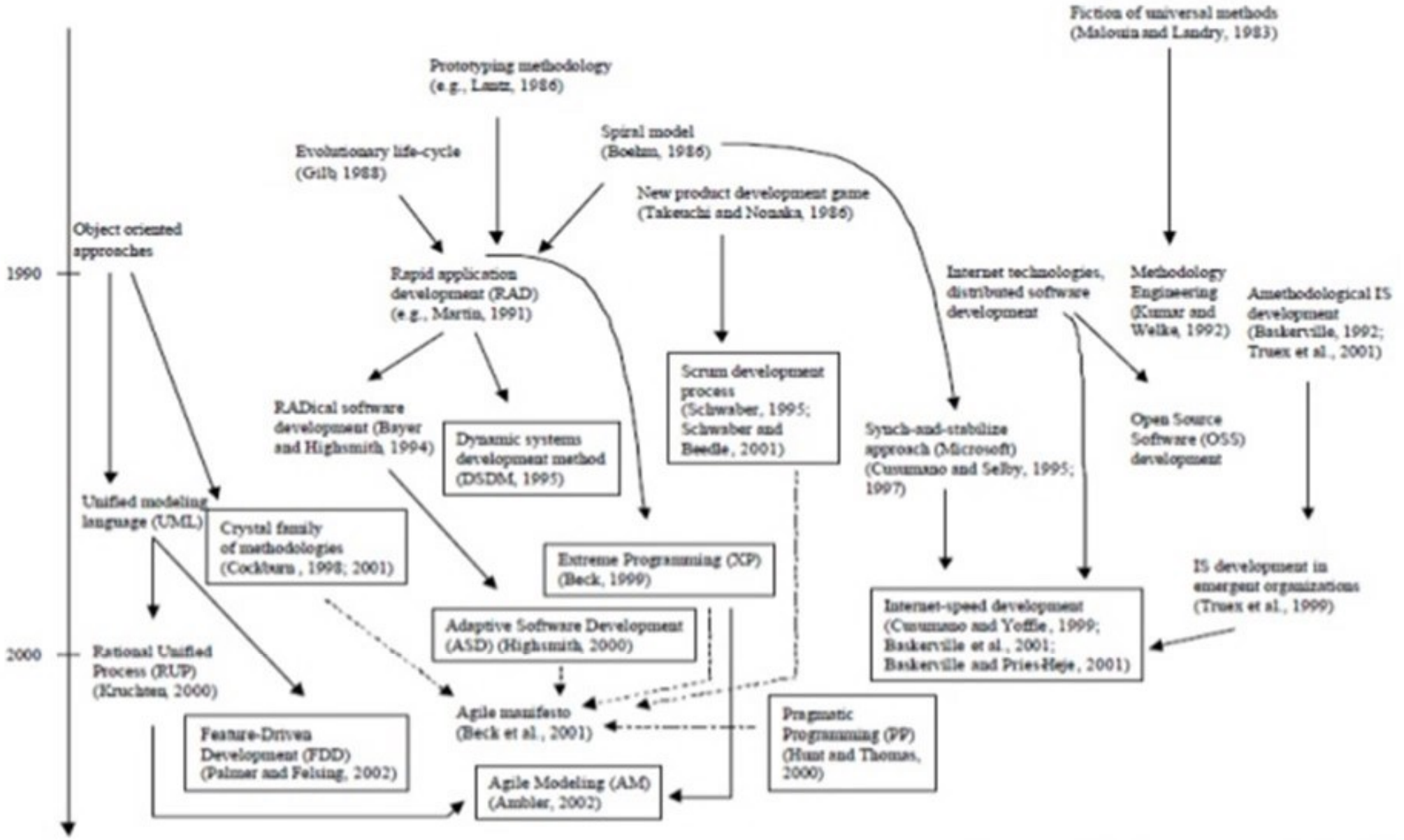
# Agile vs Traditional

- Adaptive vs Predictive;
- Requirements are static vs dynamic;
- Lightweight vs heavyweight;

# Agile – Pros / Cons

| Agile - Pros | Agile – Cons |
|---|---|
| Realistic approach, simple in nature | Not suitable for handling complex dependencies |
| Promotes teamwork | People dependent |
| Able to quickly deliver the functionalities | Expertise must present |
| Delivers working product early | Strict deadlines |
| Lightweight | Depends heavily on customer interaction |
| Minimal planning enough | Adaptability (technology) is a concern |

# Evolution of Agile Methods




**Source: Abrahamsson et al., 2003**

# Agile Methodologies – Agile Manifesto

# Agile Methodologies – Agile Manifesto

We are uncovering better ways of developing s/w –

- **Individuals and Interactions** over process and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to Change** over following a plan

While there is Value in the items on the right, we value the items on the left more.

# Agile Methodologies – Agile Manifesto

# Agile Manifesto

Agile Manifesto principles are:

- **Individuals and interactions** - in agile development, self-organization and motivation are important, as are *interactions and pair programming.*

- **Working software** - *Demo working software* is considered the *best means of communication* with the customer to understand their requirement, instead of just depending on documentation.

# Agile Manifesto – contd..

- **Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, *continuous customer interaction is very important* to get proper product requirements.

- **Responding to change** - agile development is focused on *quick responses to change* and continuous development.

# Agile Methodologies - Principles

# Agile Methodologies - Principles

## 12 Principles of Agile Software Development

1. Satisfy the customer through early and continuous delivery.
2. Welcome changing requirements, even late in development.
3. Deliver working software frequently
4. Business people and developers work together daily
5. Build projects around motivated individuals.
6. Convey information via face-to-face conversation.
7. Working software is the primary measure of progress.
8. Maintain a constant pace indefinitely.
9. Give continuous attention to technical excellence
10. Simplify: maximizing the amount of work not done
11. Teams self-organize.
12. Teams retrospect and tune behavior

# 12 Principles behind the Agile Manifesto

1. Our **highest priority** is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently** from a couple of weeks to a couple of months, with a preference to the shorter timescale.

# 12 Principles behind the Agile Manifesto

4. Business people and developers must **work together daily** throughout the project.
5. **Build projects around motivated individuals.** Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation.**

# 12 Principles behind the Agile Manifesto

7. **Working software** is the primary measure of progress.
8. Agile processes **promote sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. **Continuous attention** to technical excellence and good design enhances agility.

# 12 Principles behind the Agile Manifesto

10. **Simplicity** -the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams.**
12. At regular intervals, the **team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly (**team retrospect)**.

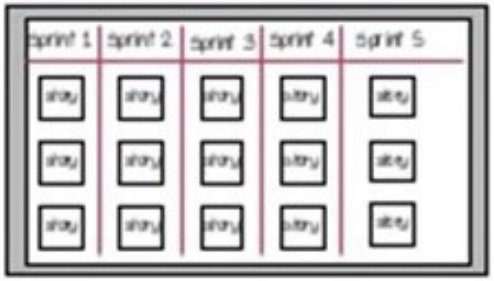
Knowledge TRAIN

# The 12 agile principles*

## 1 Satisfy the **customer**


## 2 Welcome **change**

## 3 Deliver **frequently**



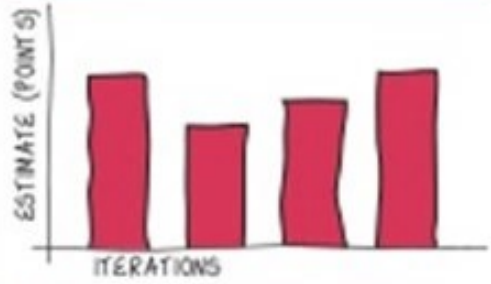

## 4 Work **together**

## 5 Trust and **support**

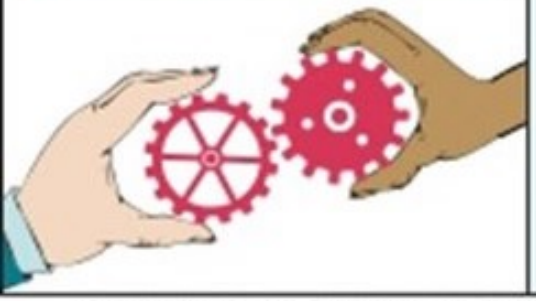
## 6 Face-to-face **conversation**

## 7 Working **software**

## 8 Sustainable **development**




## 9 Continuous **attention**



## 10 Maintain **simplicity**


## 11 Self-organizing **teams**

## 12 Reflect and **adjust**

# Agile Principles – contd..

01 Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

02 Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

03 Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

04 Business people and developers must work together daily throughout the project.

05 Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

06 Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

07 Working software is the primary measure of progress.

08 The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

09 Continuous attention to technical excellence and good design enhances agility.

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

11 The best architectures, requirements, and designs emerge from self-organizing teams.

12 At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

# References


- K.S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 2012.
- M. Cohn, Succeeding with Agile: Software Development Using Scrum, Addison-Wesley, 2009
- S.W. Ambler, M. Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press, 2012.
- Chetankumar Patel, Muthu Ramachandran, Story Card Maturity Model (SMM): A Process Improvement Framework for Agile Requirements Engineering Practices, Journal of Software,Academy Publishers, Vol 4, No 5 (2009), 422-435, Jul 2009.
- Kevin C. Desouza, Agile information systems: conceptualization, construction, and management, Butterworth-Heinemann, 2007
- K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, 2nd Edition, Addison-Wesley, 2004.

# Thank you!