1151CS112 – OBJECT ORIENTED SOFTWARE ENGINEERING

UNIT-1

Course Category : Program Core

Credits : 3

Slot : S1 & S11

Semester : Summer

Academic Year : 2022

Faculty Name : Saranya R

School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of
Science and Technology

Prerequisite and Related Courses

Prerequisite Courses:

Sl.No	Course Code	Course Name
1	1151CS107	Database Management System

Related Courses

Sl.No	Course Code	Course Name
1	1151CS201	Mobile Application Development

Course Outcomes

CO Nos.	Course Outcomes	K Level
CO1	Summarize about software development process models	K2
CO2	Estimate the software project based on project planning and its constraints.	K2
CO3	Construct and Sketch the UML diagrams for given scenario.	K3
CO4	Explain the software design heuristics for quality improvement.	K2
CO5	Discuss on different types of testing strategies.	K2

Syllabus

Unit	Course Outline
1	Introduction
2	Planning & Scheduling
3	Analysis
4	Design
5	Implementation, Testing & Maintenance

Learning Resources

S.No	Text Book
1	Roger. S. Pressman and Bruce R. Maxim, "Software Engineering – A Practitioner's Approach", seventh Edition, McGraw Hill, 2015.
2	Ian Sommerville, "Software Engineering", eighth edition, Pearson Education, New Delhi, 2011.
3	Ali Bahrami, "Object Oriented Systems Development" 1st Edition, The McGraw-Hill Company, 1999
4	Craig Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition), Pearson Education, 2008.

Learning Resources

S.No	Reference Book
1	Fairley R, “Software Engineering Concepts”, second edition, Tata McGraw Hill,New Delhi, 2003.
2	Jalote P, “An Integrated Approach to Software Engineering”, third edition, Narosa Publishers, New Delhi, 2013.
3	Grady Booch, James Rumbaugh, Ivar Jacobson - "the Unified Modeling Language User Guide" - Addison Wesley, 1999.
4	Bill Barczewski, Richard D. Stutz,” Software Engineering Project Management”, Wiley India Edition, IEEE computer society, 2007.

UNIT- I Introduction

- Introduction to Software Engineering
- Software Development process models
- Agile Development
- Project & Process
- Process & Project metrics
- Project management
- Object Oriented concepts, Principles & Methodologies

Introduction to Software Engineering

- Introduction
- SoftwareEngineering
- Software Products
- Why software important
- Application of software
- Features of software
- Software Paradigms
- Software Engineering
- Importance & Challenges of software Engineering
- Professional and ethical responsibility
- Layered Technology

Introduction

➢ Software Engineering=Software + Engineering.

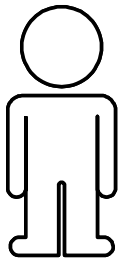
❑ **Software**

- Software is considered to be a collection of executable programming code, associated libraries and documentations.
- Software, when made for a specific requirement is called software product.

❑ **Engineering**

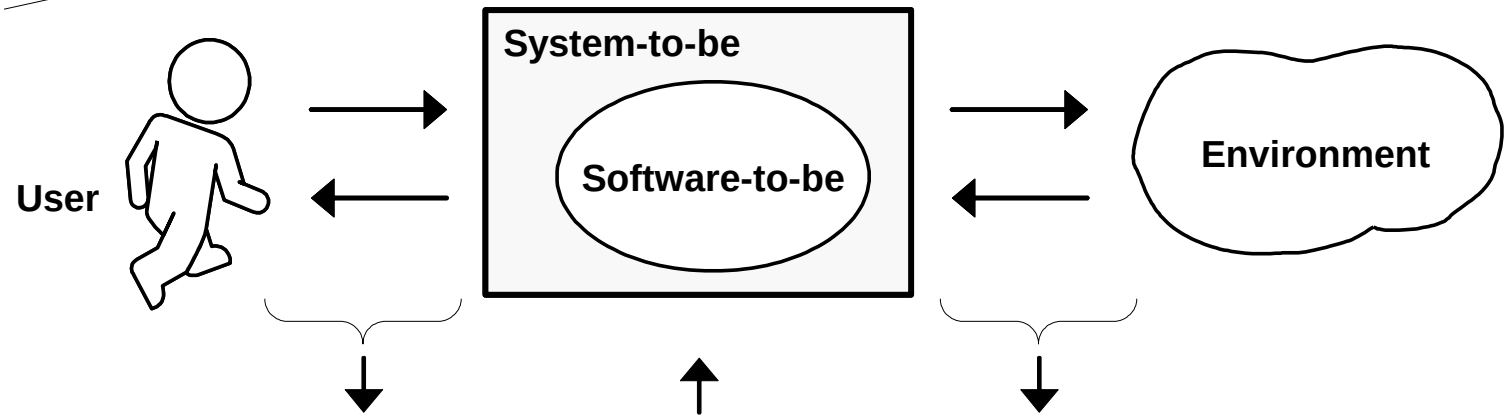
- Engineering is all about developing products, using well-defined, scientific principles and methods.

Role of Software Engineering



Customer:

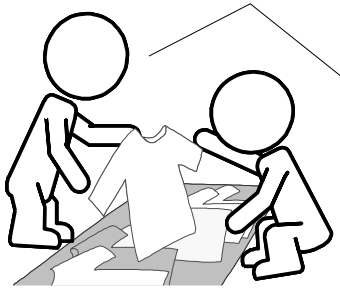
Requires a computer system to *achieve some business goals* by user interaction or interaction with the environment in a specified manner



Software Engineer's task:

To ***understand how*** the system-to-be needs to interact with the user or the environment so that customer's requirement is met and ***design*** the software-to-be

May be the same person



Programmer's task:

To ***implement*** the software-to-be designed by the software engineer

Software products

- **Generic products**

  Stand-alone systems that are marketed and sold to **any customer** who wishes to buy them.

  Examples – PC software such as editing, graphics programs,project management tools; CAD software; software for specific markets such as appointments systems for dentists.

- **Customized products**

  Software that is commissioned by **a specific customer** to meet their own needs.

  Examples –embedded control systems, air traffic control software, traffic monitoring systems.

Why Software is Important?

- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment,)
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

Software Applications

1.System software: such as compilers, editors, file management utilities

2. Application software: stand-alone programs for specific needs.

3.Engineering/scientific software: Characterized by " number crunching"algorithms. such as automotive stress analysis, molecular biology, orbital dynamics etc

4.Embedded software resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)

Software Applications

5.Product-line software focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)

6. WebApps (Web applications) network centric software. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.

7. AI software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing

Features of Software

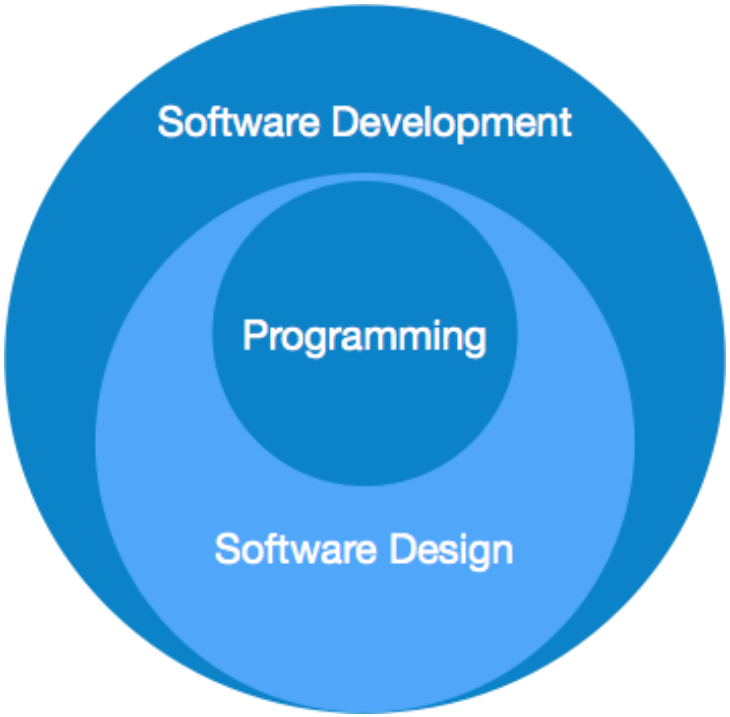
- Software is developed or engineered, it is not manufactured in the classical sense which has quality problem.
- Software doesn't "wear out." but it deteriorates (due to change).
- most software continues to be custom-built.

Attributes of good software

- Maintainability
- Dependability
- Efficiency
- Usability

Software Paradigms

- Software paradigms refer to the methods and steps,which are taken while designing the software
- This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied.



Software Paradigms

Software Paradigms consists of

- ✓ Requirement gathering
- ✓ Software design
- ✓ Programming

Software Design Paradigm

This paradigm is a part of Software Development and includes

- ✓ Design
- ✓ Maintenance
- ✓ Programming

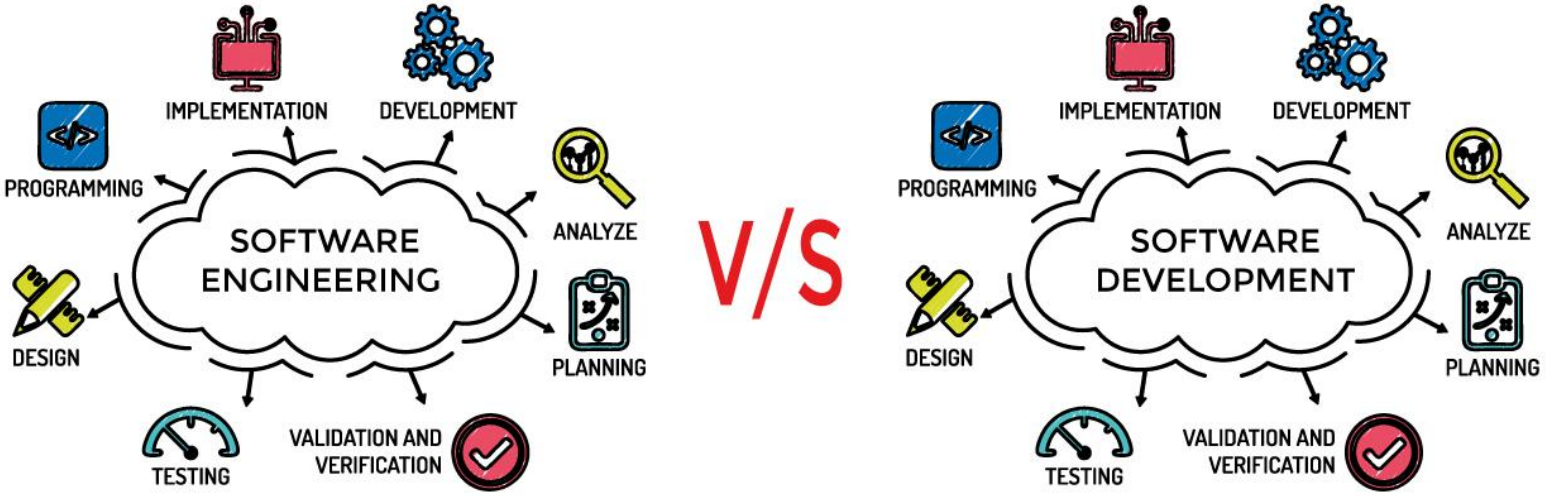
Programming Paradigm

This paradigm is related closely to programming aspect of software development.

- ✓ Coding
- ✓ Testing
- ✓ Integration

Software Engineering

Software Engineering is a systematic approach to develop and maintain a software product in a cost effective and efficient way.



Importance of Software Engineering

- More and more, individuals and society rely on advanced software systems.
- We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.
- For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

Challenges in Software Engineering

What are the key challenges facing software engineering?

- Coping with legacy systems, coping with increasing diversity and coping with demands for reduced delivery times

- Legacy systems

  Old, valuable systems must be maintained and updated

- Heterogeneity

  Systems are distributed and include a mix of hardware and software

- Delivery

  There is increasing pressure for faster delivery of software

Professional & Ethical responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals
- Ethical behaviour is more than simply upholding the law.

Difference in Software Engineering

- What is the difference between software engineering and computer science?

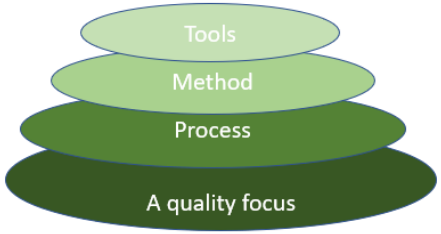
Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

- What is the difference between software engineering and system engineering?

System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

Layered Technology

- Software Engineering is a layered technology that rest on the quality.



- Process layer-process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology
- Software engineering methods provide the technical how to for building software
- Method encompass tasks that include requirement analysis, design, construction, testing and maintenance
- **Tools** provide automated or semi-automated support for the process and methods

Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created.

- Purpose of process is to deliver software in a timely manner and with sufficient quality.

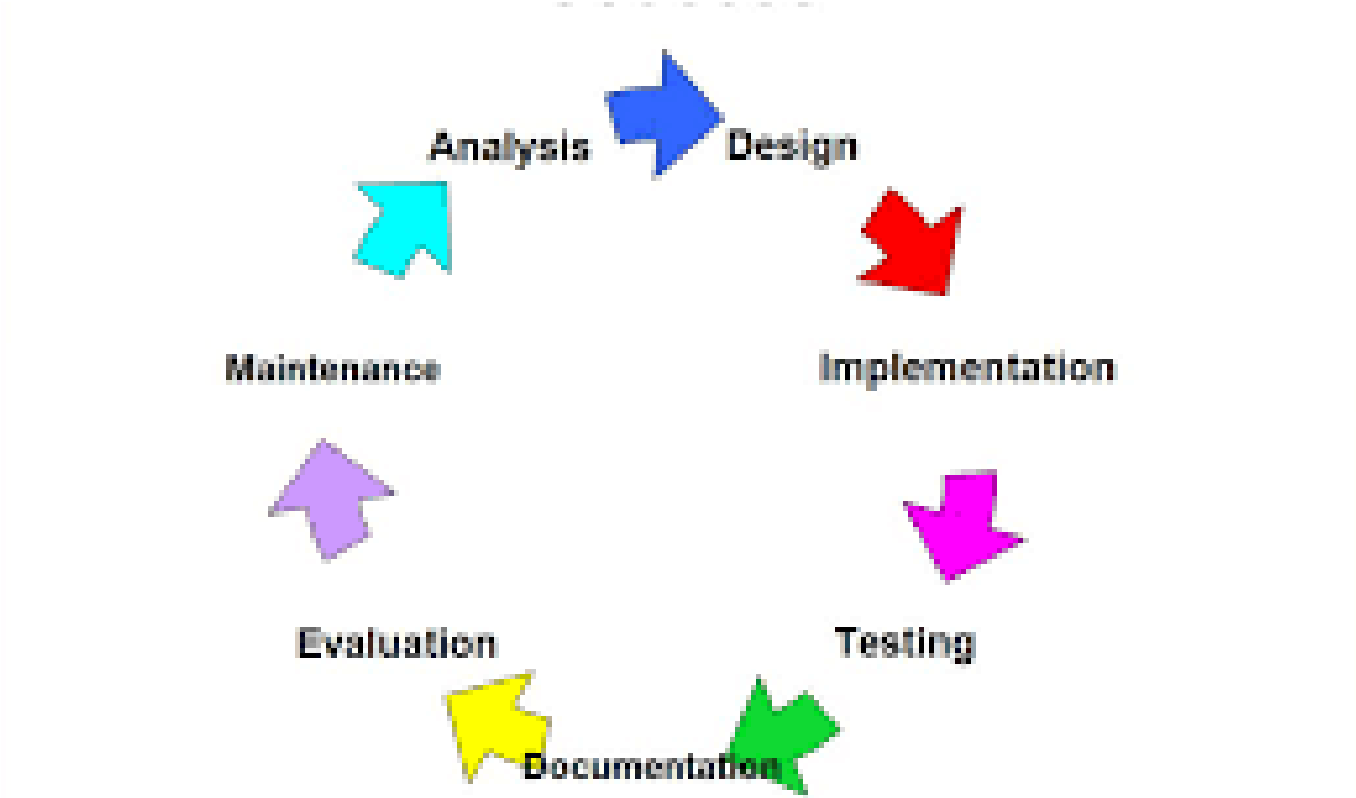
✓ Specification - what the system should do and its development constraints

✓ Development - production of the software system

✓ Validation - checking that the software is what the customer wants

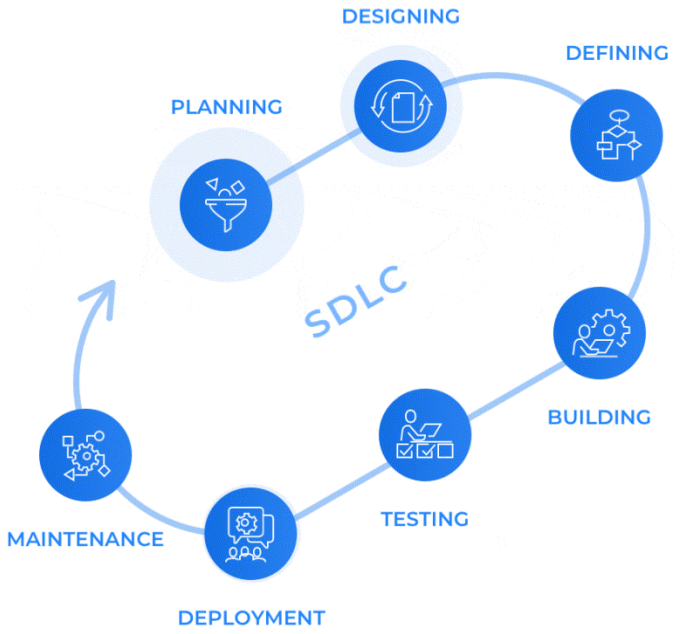
✓ Evolution - changing the software in response to changing demands.

Software Development Process



Software Development process models

- **Software Process**
- **What / who / why is Process Models?**
- **Software Development Process model**
- **Types of Software Process Model**
  - ✓ Waterfall model
  - ✓ V model
  - ✓ Incremental model
  - ✓ Iterative model
  - ✓ RAD model
  - ✓ Spiral model
  - ✓ Prototype model
  - ✓ Agile model
  - ✓ Bigbang Model

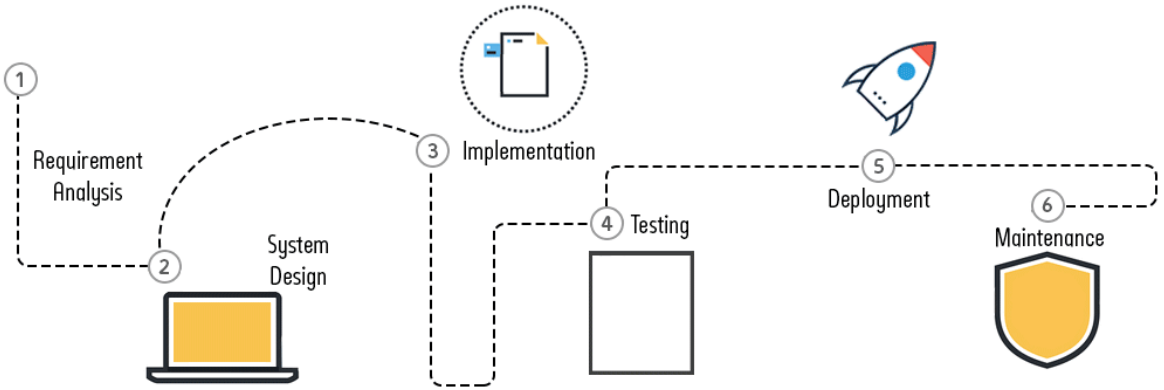


What / who / why is Process Models?

What: Go through a series of predictable steps--- a road map that helps you create a timely, high-quality results.

Who: Software engineers and their managers, clients also. People adapt the process to their needs and follow it.

Why: Provides stability, control, and organization to an activity that can if left uncontrolled, become quite chaotic.



What / who / why is Process Models?

What Work products: Programs, documents, and data.

What are the steps: The process you adopt depends on the software that you are building. One process might be good for aircraft avionic system, while an entirely different process would be used for website creation.

How to ensure right: A number of software process assessment mechanisms that enable us to determine the maturity of the software process. However, the quality, timeliness and long-term viability of the software are the best indicators of the efficacy of the process you use.

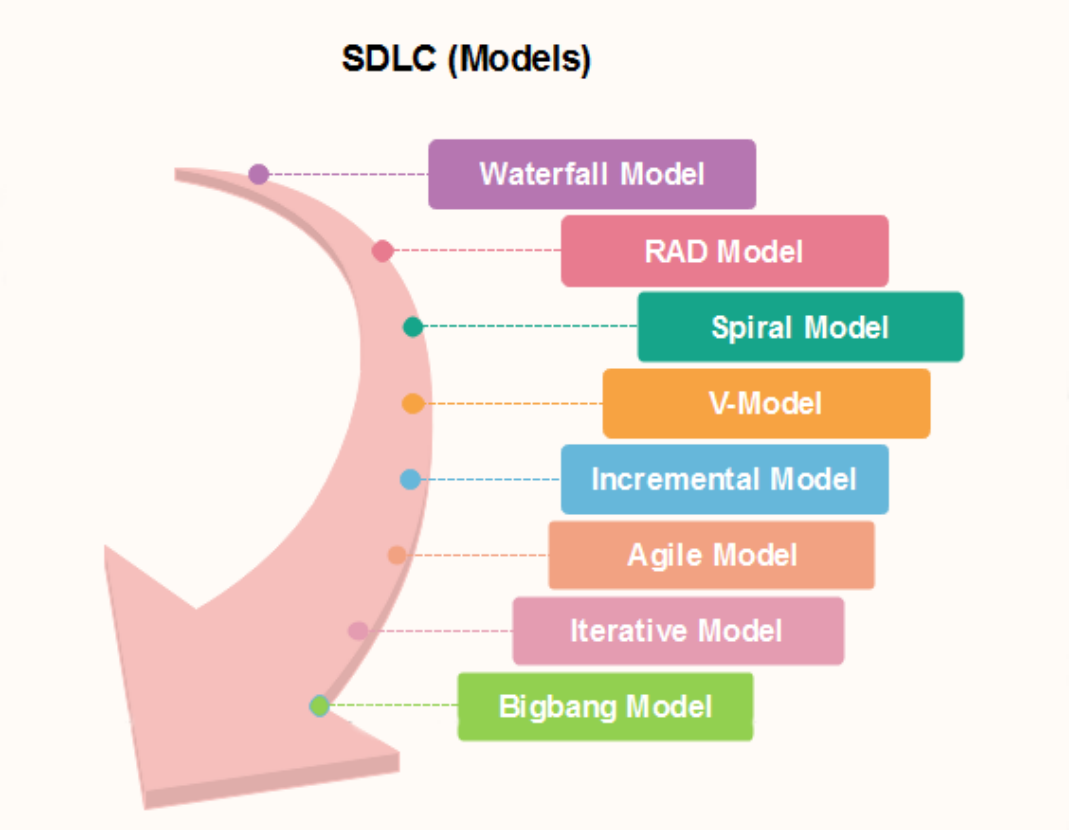
Software Development process models

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created.
- One of the basic notions of the software development process is SDLC models which stands for Software Development Life Cycle models.
- There are many development life cycle models that have been developed in order to achieve different required objectives.
- The models specify the various stages of the process and the order in which they are carried out.

Types SDLC Models

Types of software Development process model

- Waterfall model
- V model
- Incremental model
- Iterative model
- Spiral model
- Prototype model
- RAD model
- Agile model
- Bigbang model

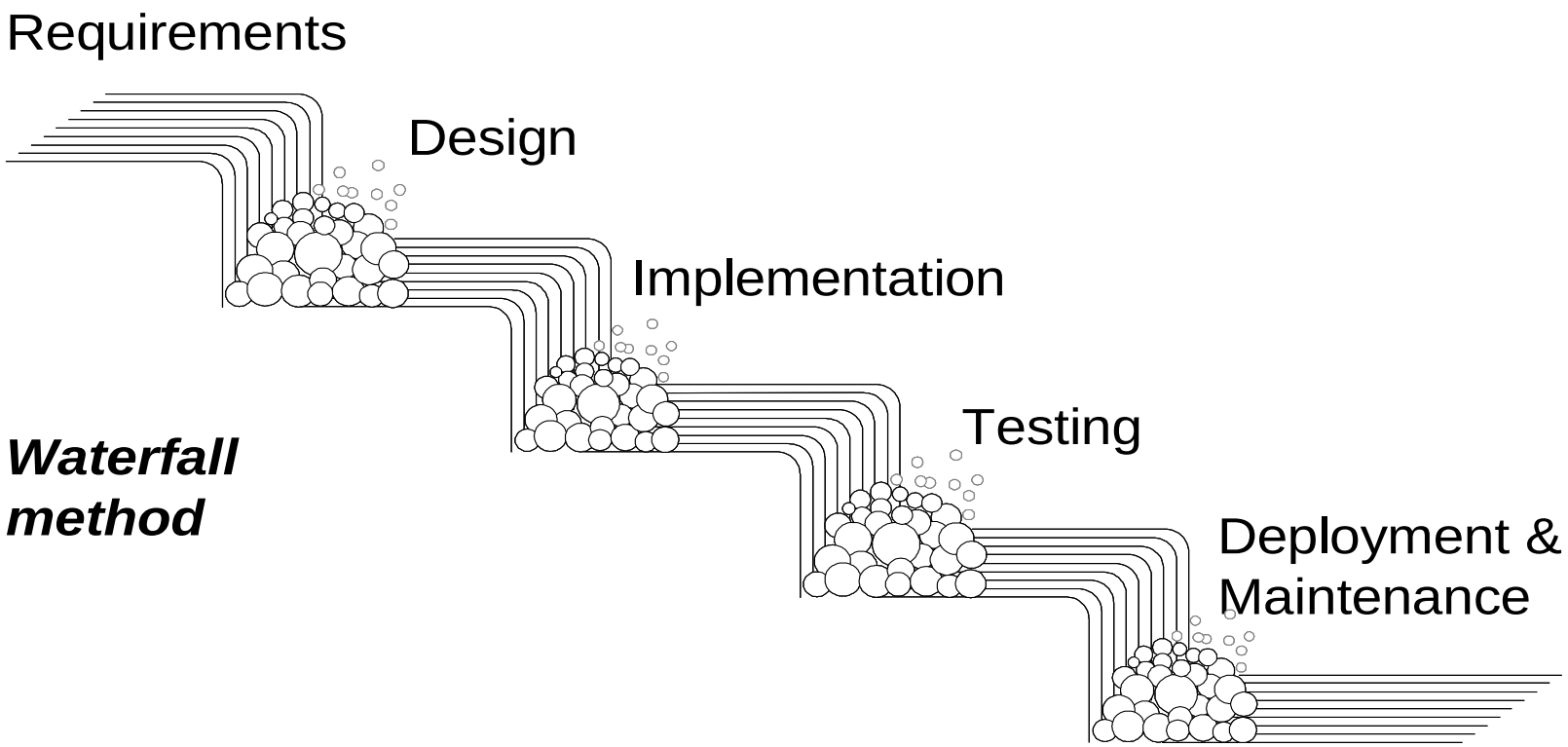


Waterfall model

- It is a Oldest model and is known as linear sequential model or classic life cycle model

- The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks.

Waterfall model



Each activity confined to its "phase".

Unidirectional, no way back;

finish this phase before moving to the next

Waterfall model

❑ Communication

- ✓ All the possible requirements of the system developed are captured in this phase.
- ✓ Communication between customer and developer.

❑ Planning

- ✓ It consist of complete estimation, scheduling for project development.

❑ Modelling

- ✓ Defines overall system architecture.
- ✓ Helps in specifying hardware and software
- ✓ Architecture creates low level and high level design

Waterfall model

❑ Construction

- ✓ Consist of code generation and testing.
- ✓ Coding implements the design details using an appropriate programming language.
- ✓ Testing the flow of coding and checking the functionality
- ✓ Testing is done by unit testing, Integration testing and system testing
- ✓ Unit testing-testing individual component
- ✓ Integration testing-individual units are combined and tested
- ✓ System testing- testing that validates the complete and fully integrated software product.

❑ Deployment

- ✓ Delivering the product to the customer and taking feedback from them.
- ✓ Software must be maintained.

Waterfall model

Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall model

Disadvantages

- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.

Waterfall model

When to use 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.

V model

V Model, A variation of waterfall model depicts the relationship of quality assurance actions to the actions associated with communication, modeling and early code construction activates

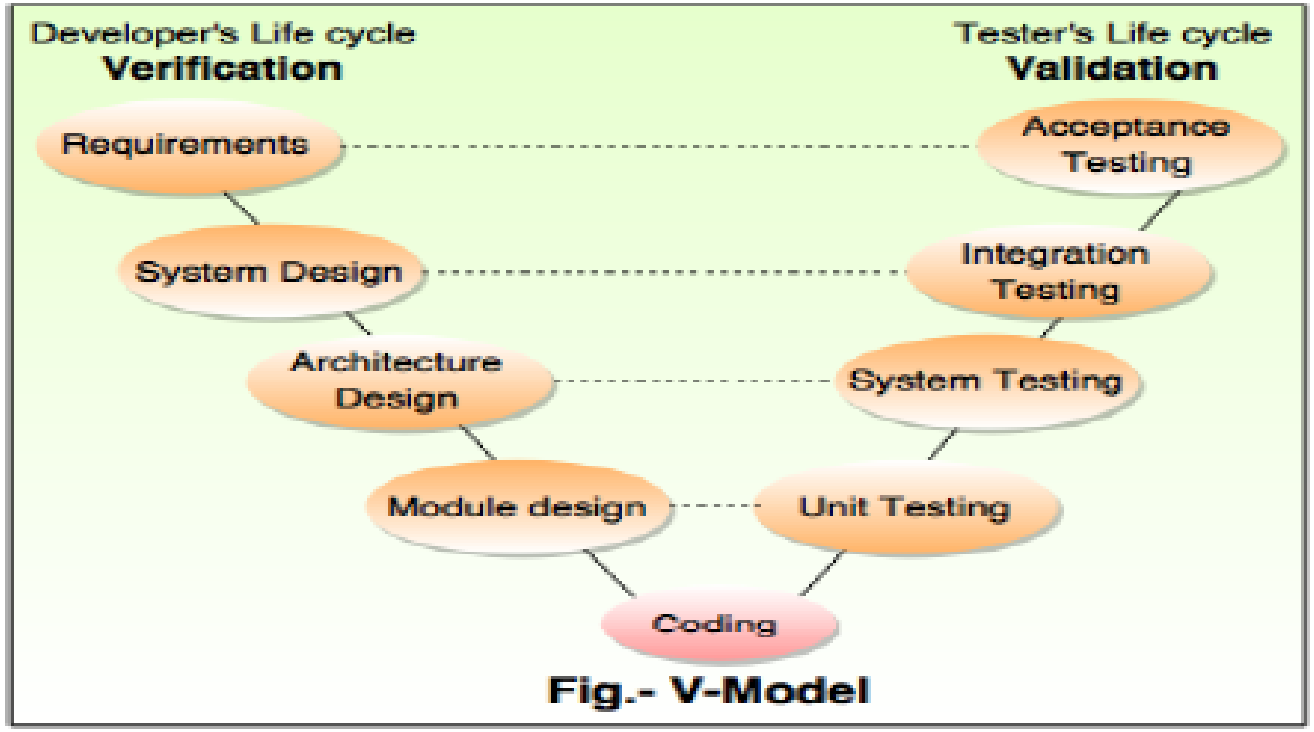


Fig.- V-Model

V model

- **The V-Model** demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
- It is also known as **verification and validation** model.
- Verification-it is the process of evaluation of the product development phase to find whether specified requirements are meet. It answers the question "**Am I building a product right?**"
- Validation-testing done by executing code and also testing determine whether software meets the customer expectations and requirements. It answers the question "**Am I building a right product?**"

V model

Advantages

- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Each phase has specific deliverables and a review process

Disadvantages

- High risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.

Incremental Model

- The incremental model combines elements of linear and parallel process flows.
- Incremental model in software engineering is one which combines the elements of waterfall model which are then applied in a iterative model

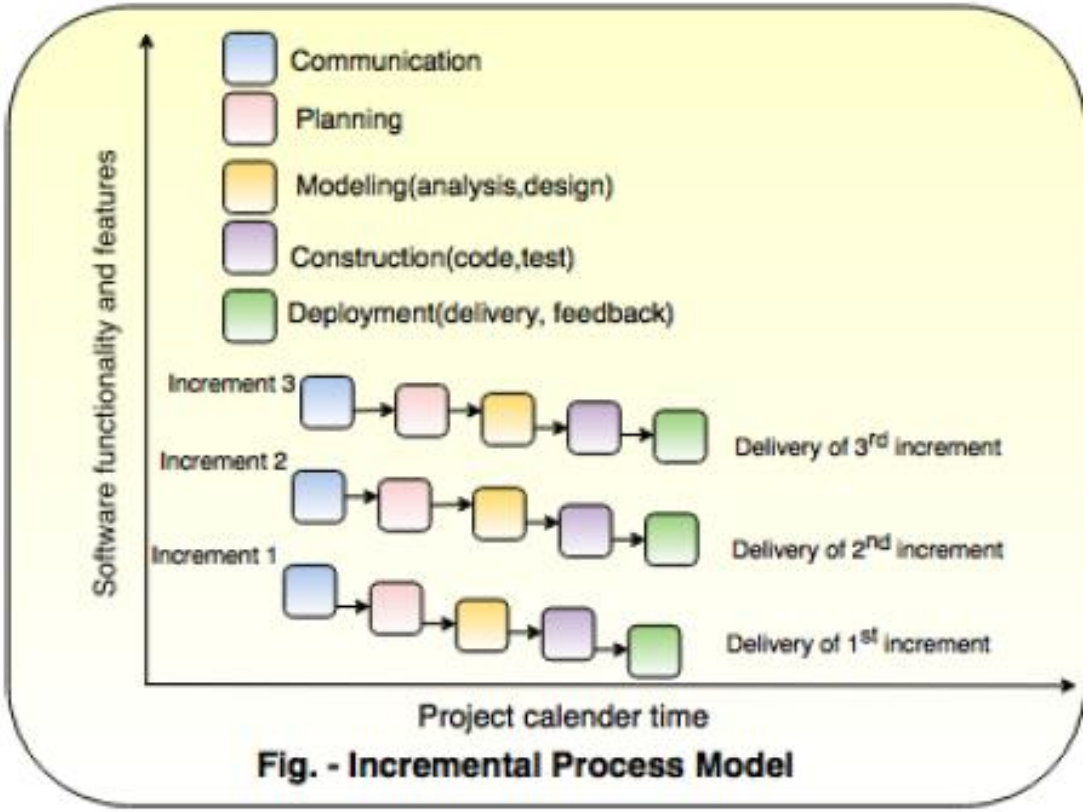


Fig. - Incremental Process Model

Incremental Model

- The first increment is often a **core product** with many supplementary features. Users use it and evaluate it with more modifications to better meet the needs.
- **Example:**

Word Processing Software

- ✓ 1st Increment- Document Processing facility
- ✓ 2nd Increment- More sophisticated document processing &file management facility
- ✓ 3rd Increment-Spelling&Grammer checking facility

Incremental Model

Advantages

- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Lowers initial delivery cost
- Risk analysis is better

Disadvantages

- Not suitable for small project.
- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall

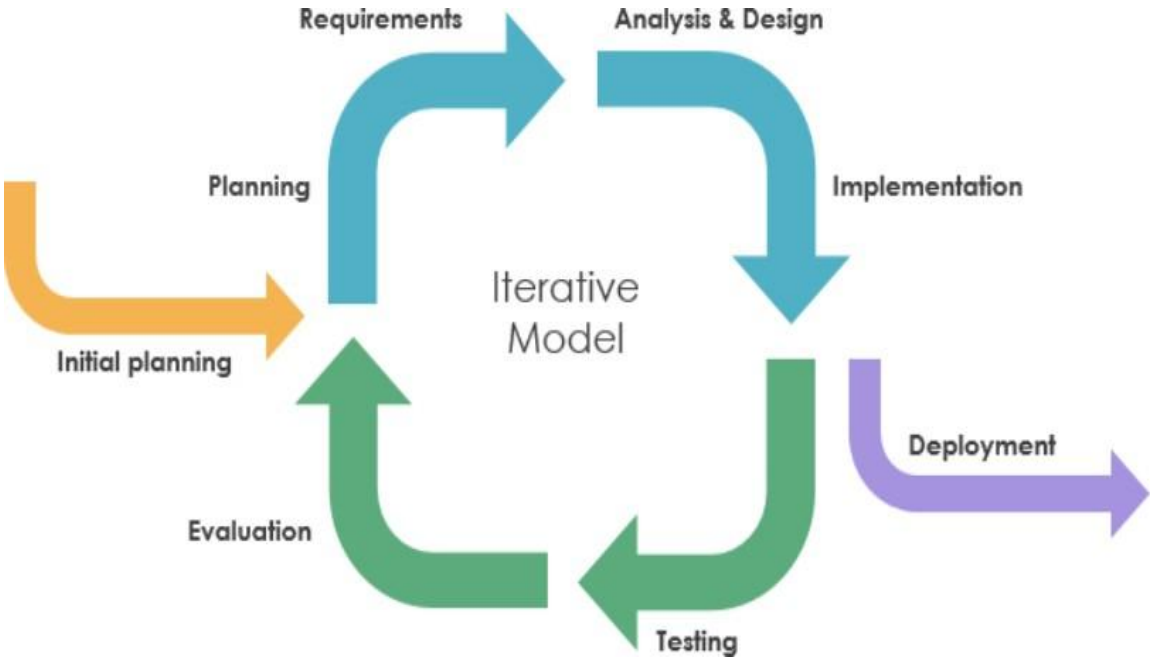
Incremental Model

When to use Incremental Model

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goal

Iterative model

- Iterative method, the concept of incremental development will also often be used liberally and interchangeably, which describes the incremental alterations made during the design and implementation of each new iteration.



Iterative model

- The iterative approach begins by specifying and implementing just part of the software, which can then be reviewed and prioritized in order to identify further requirements.

- This iterative process is then repeated by delivering a new version of the software for each iteration.

When to use iterative model?

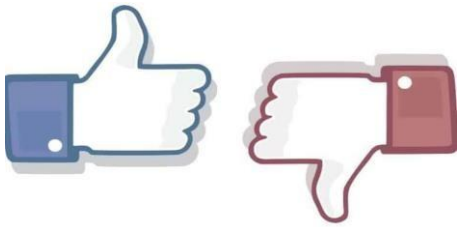
- Requirements of the complete system are clearly defined and understood.
- When the project is big.
- Major requirements must be defined; however, some details can evolve with time.

Iterative model

Advantages

- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- It supports changing requirements.
- Testing and debugging during smaller iteration is easy.

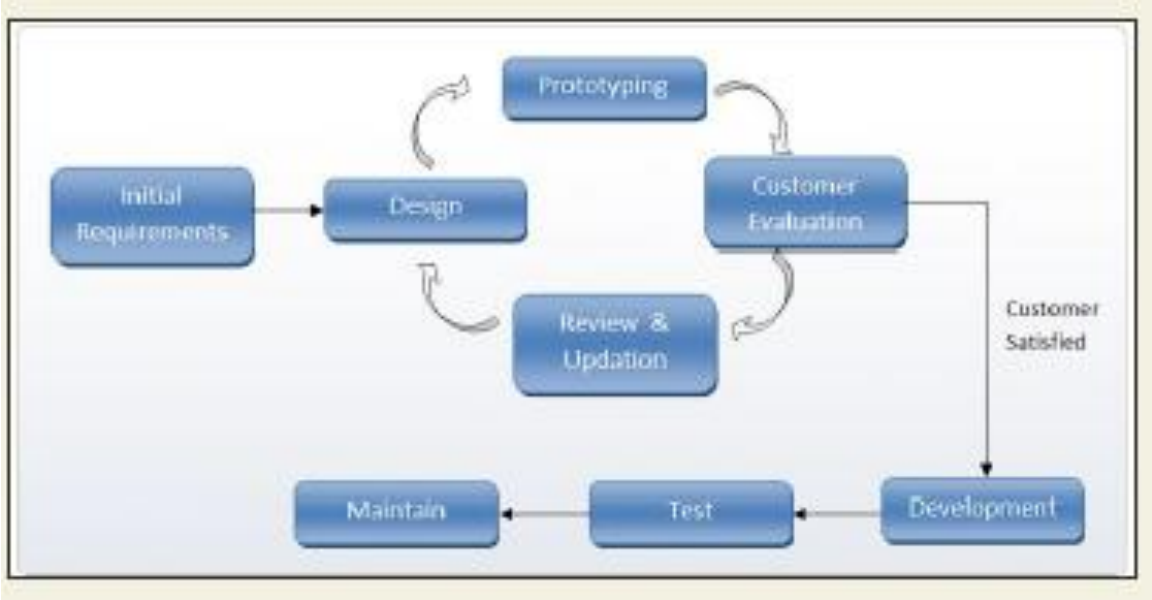
Disadvantages



- Not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.

Prototype model

The **Prototype model** is one of the software development life cycle models in which a **prototype is built with minimal requirements**, which is then tested and modified based on the **feedback received** from the client until a final prototype with desired functionalities gets created. This final prototype also acts as a base for the final product.



Prototype model

Advantages

- Quick client feedback is received which speeds up the development process.
- Also, it helps the development team to understand the client's needs.
- Developed prototypes can be used later for any similar projects.
- Any missing functionality and any error can be detected early.
- It is useful when requirements are not clear from the client's end, even with limited requirements, the development team can start the development process.

Prototype model

Disadvantages

- It is a time-consuming process
- This method involves too much client interaction and involvement.
- Prototyping tools are expensive

When to use prototype model?

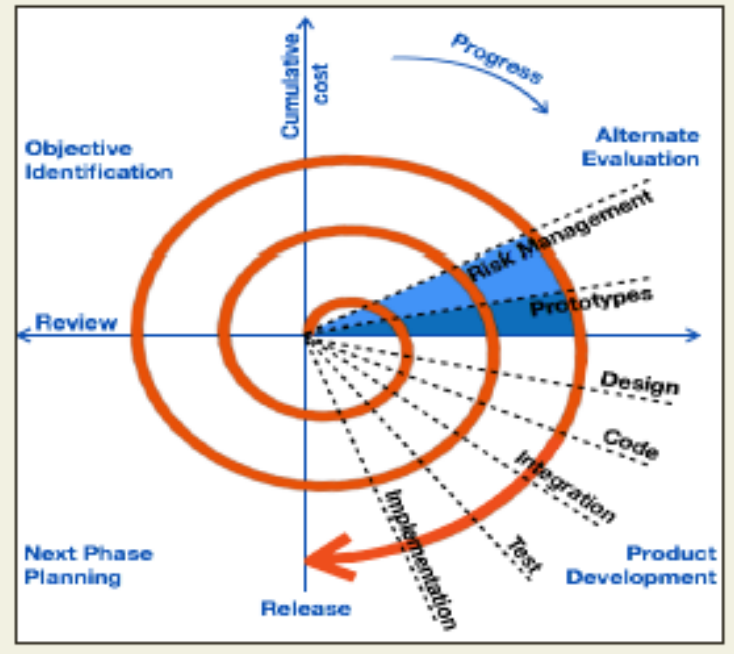
Customer defines a set of general objectives but does not identify detailed requirements for functions and features.

(Or)

Developer may be unsure of the efficiency of an algorithm, the form that human computer interaction should take

Spiral model

- The spiral model is similar to the incremental model, with more emphasis placed on risk analysis.
- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
- A software project repeatedly passes through these phases in iterations (called Spirals in this model).



Spiral model

Advantages

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date

Disadvantages

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.

Doesn't work well for smaller projects.

Spiral model

When to Use Spiral Model?

- When costs and risk evaluation is important.
- For medium to high-risk projects.
- Users are unsure of their needs.
- Requirements are complex.
- New product line.
- Significant changes are expected (research and exploration).

Bigbang model

- Big-Bang is the SDLC(Software Development Life cycle) model in which no particular process is followed.
- Generally this model is used for small projects .It is specially useful in academic projects.
- The development of this model begins with the required money and efforts as an input.

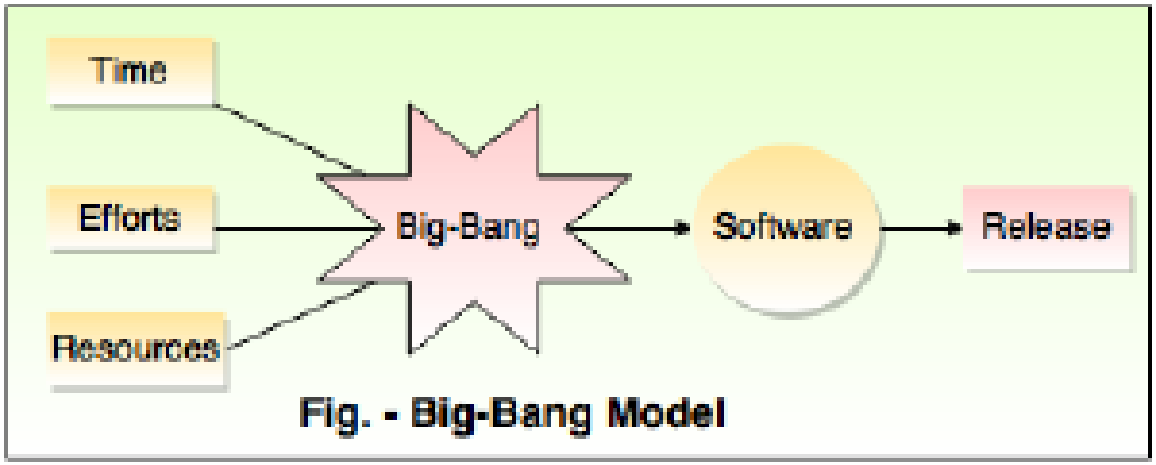


Fig. - Big-Bang Model

Bigbang model

Advantages

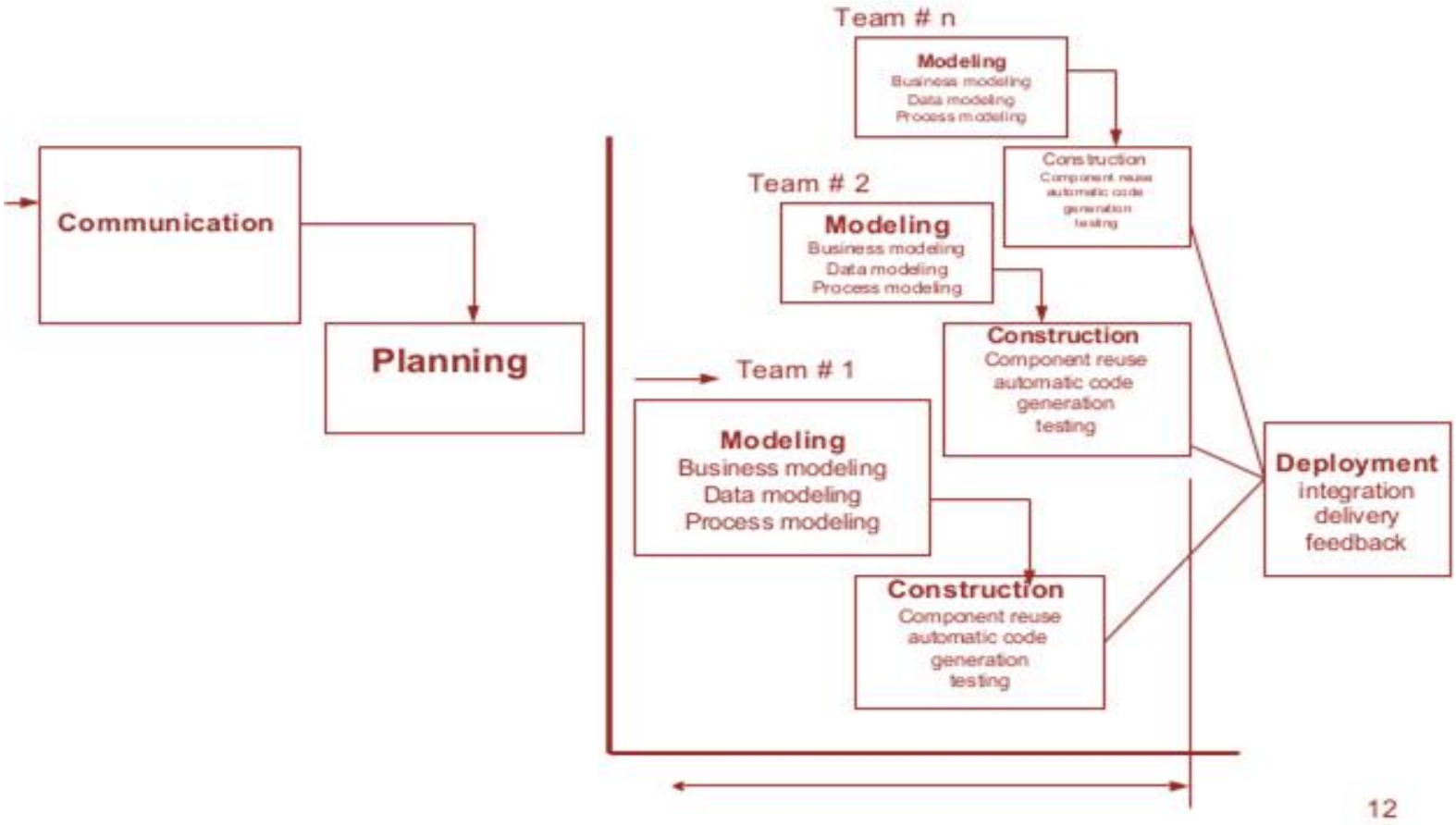
- Big-Bang model is a simple model.
- It needs little planning.
- It is simple to manage. It needs just a few resources to be developed.
- It is useful for students and new comers.

Disadvantages

- It is a very high risk model.
- This model is not suitable for object oriented and complex projects.
- Big-Bang is poor model for lengthy and in-progress projects.

RAD model

RAD which is abbreviated as Rapid Application Development Model, is based on the concepts of both iterative and prototyping development model.



RAD model

When to use RAD model?

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

RAD model

Advantages

- Reusability of components makes or speeds up the development and reduces the time that it needs for developing a product.
- Large projects can be done easily through the RAD model.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

Disadvantages

- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD

Agile Development

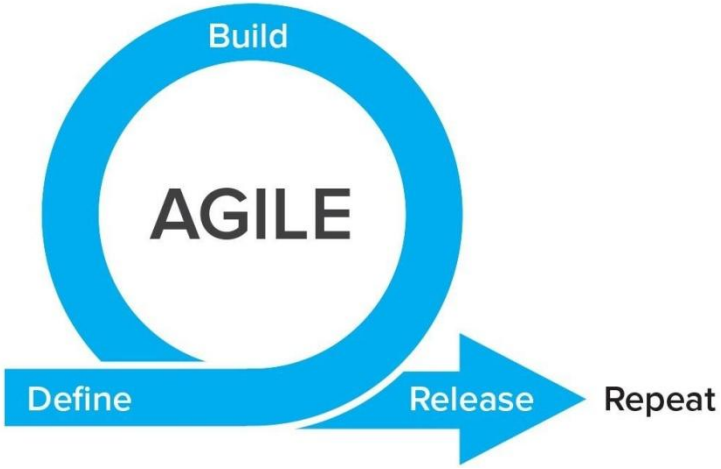
- What is Agile?
- Agile Development
- Characteristics of Agility
- Agile manifesto
- Agile principles
- Human Factors
- Agile Methodology
  - ✓ XP(Extreme Programmin
  - ✓ SCRUM
  - ✓ Crystal
  - ✓ Dynamic Software Development Method (DSDM)
  - ✓ Feature Driven Development (FDD)
  - ✓ Lean Software Development
  - ✓ Adaptive Software Development(ASD)

Agile Development

What is Agile?

- Agile means ability to create and respond to change.

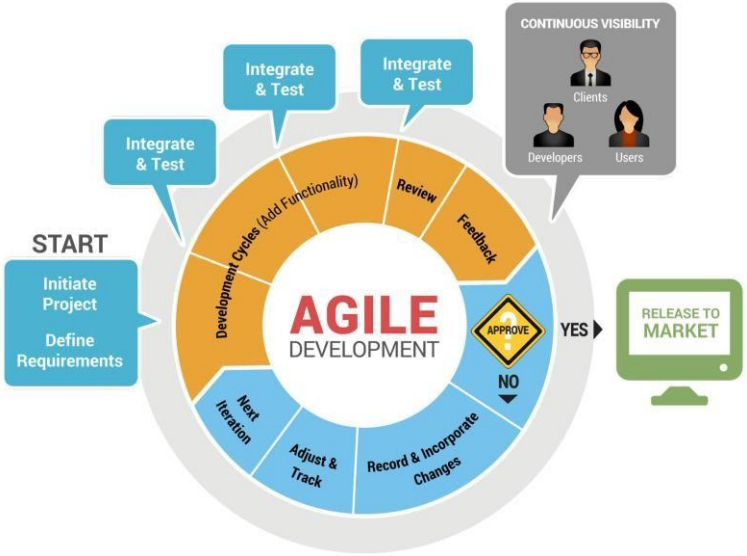
- Agile software Engineering combines a philosophy and a set of development guidelines



Agile Development

Agile software Development

- Agile development process can deliver successful system quickly.
- Agile development is a different way of managing IT development teams and projects
- Agile development stresses continuous communication and collaboration among developers and customers.



Agile Development

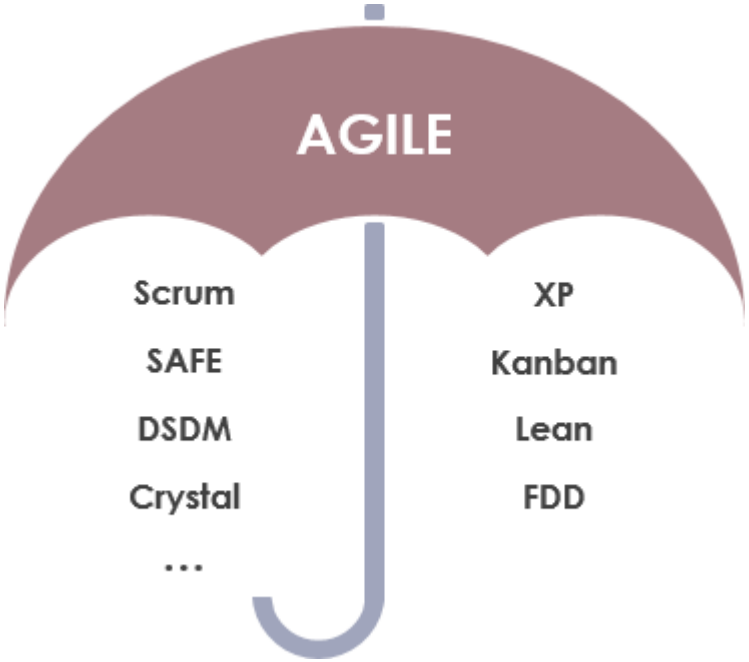
Characteristics of Agility

- Agility in Agile Software Development focuses on the culture of the whole team with multi-discipline, cross-functional teams that are empowered and selforganizing.
- It fosters shared responsibility and accountability.
- Facilitates effective communication and continuous collaboration.
- The whole-team approach avoids delays and wait times.
- Frequent and continuous deliveries ensure quick feedback that in in turn enable the team align to the requirements.
- Collaboration facilitates combining different perspectives timely in implementation, defect fixes and accommodating changes.
- Progress is constant, sustainable, and predictable emphasizing transparency

Agile Development

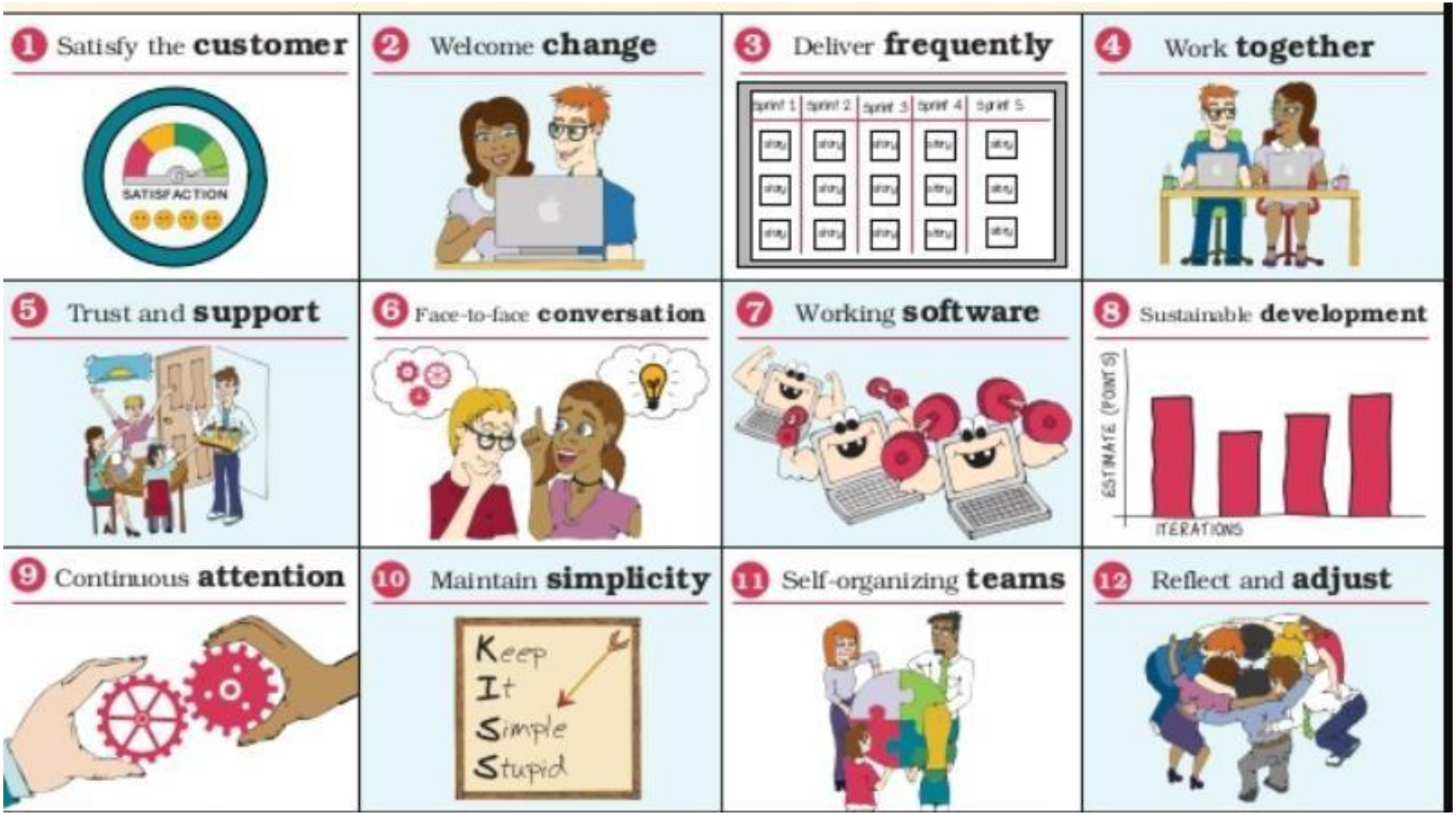
Agile manifesto

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan



Agile Development

Agile Principles



Agile Development

Human Factors in Agile Team

- Focus on talent & skills of individual
- Key traits must exist among the people on agile team

1.Competence

2.Common Focus.

3.Collaboration.

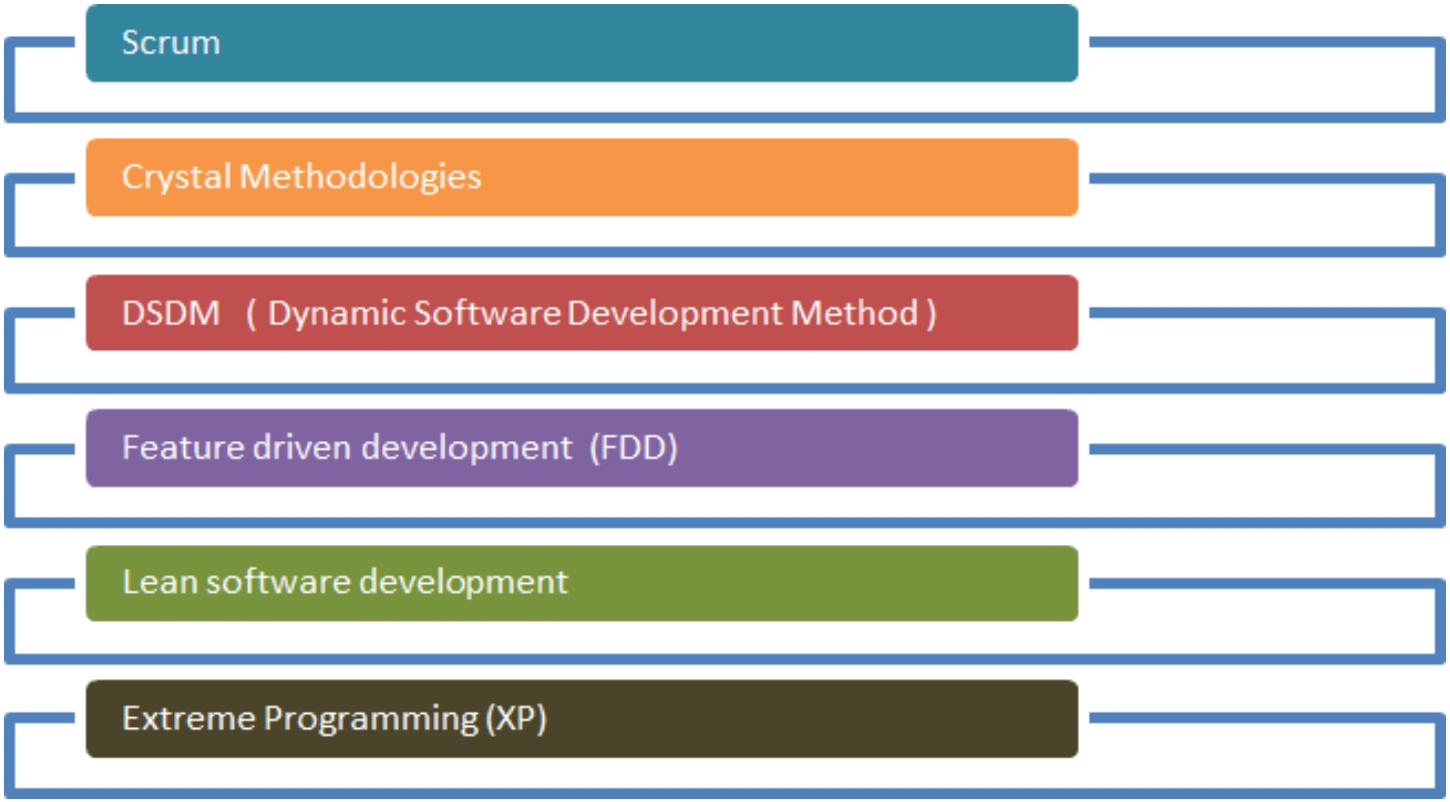
4.Decision making ability.

5.Fuzzy problem solving ability

6.Mutual trust respect

7.Self Organization

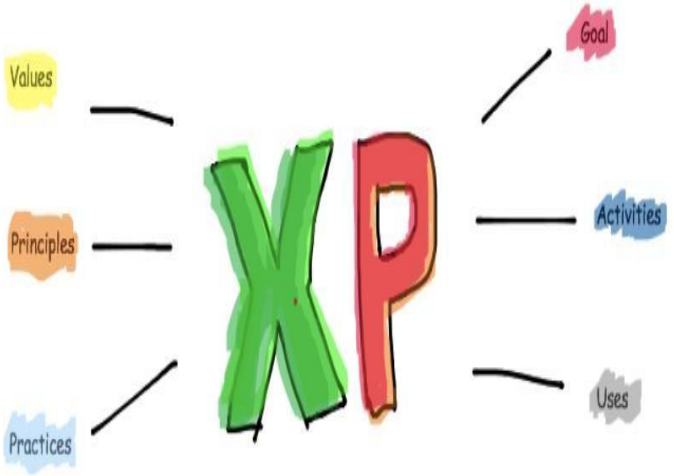
Agile Methodology



XP(Extreme Programming)

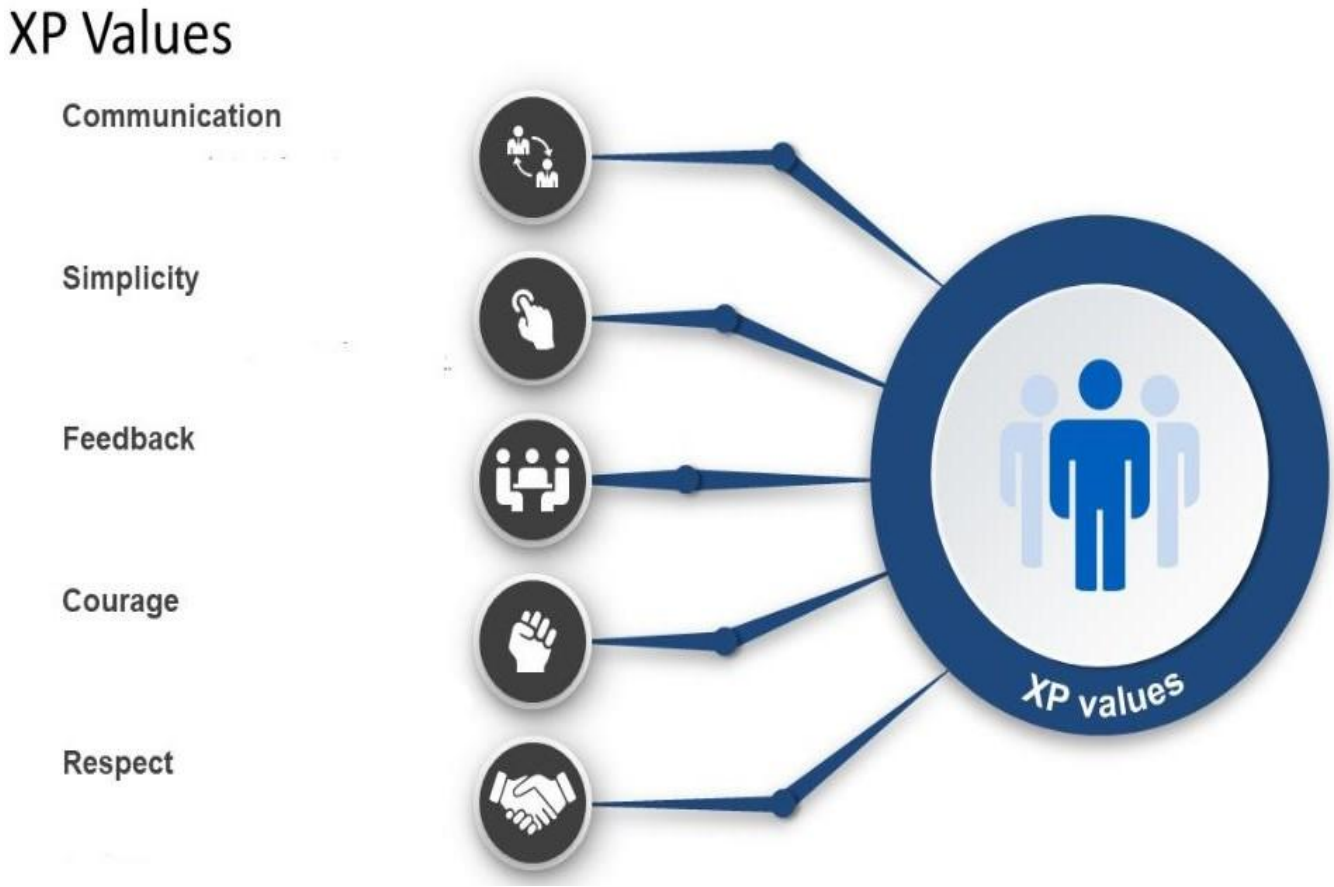
XP(Extreme Programming)

- Extreme Programming provides values and principles to guide the team behavior.
- The team is expected to self-organize.
- Extreme Programming provides specific core practices where
  - ✓ Each practice is simple and self-complete.
  - ✓ Combination of practices produces more complex and emergent behavior

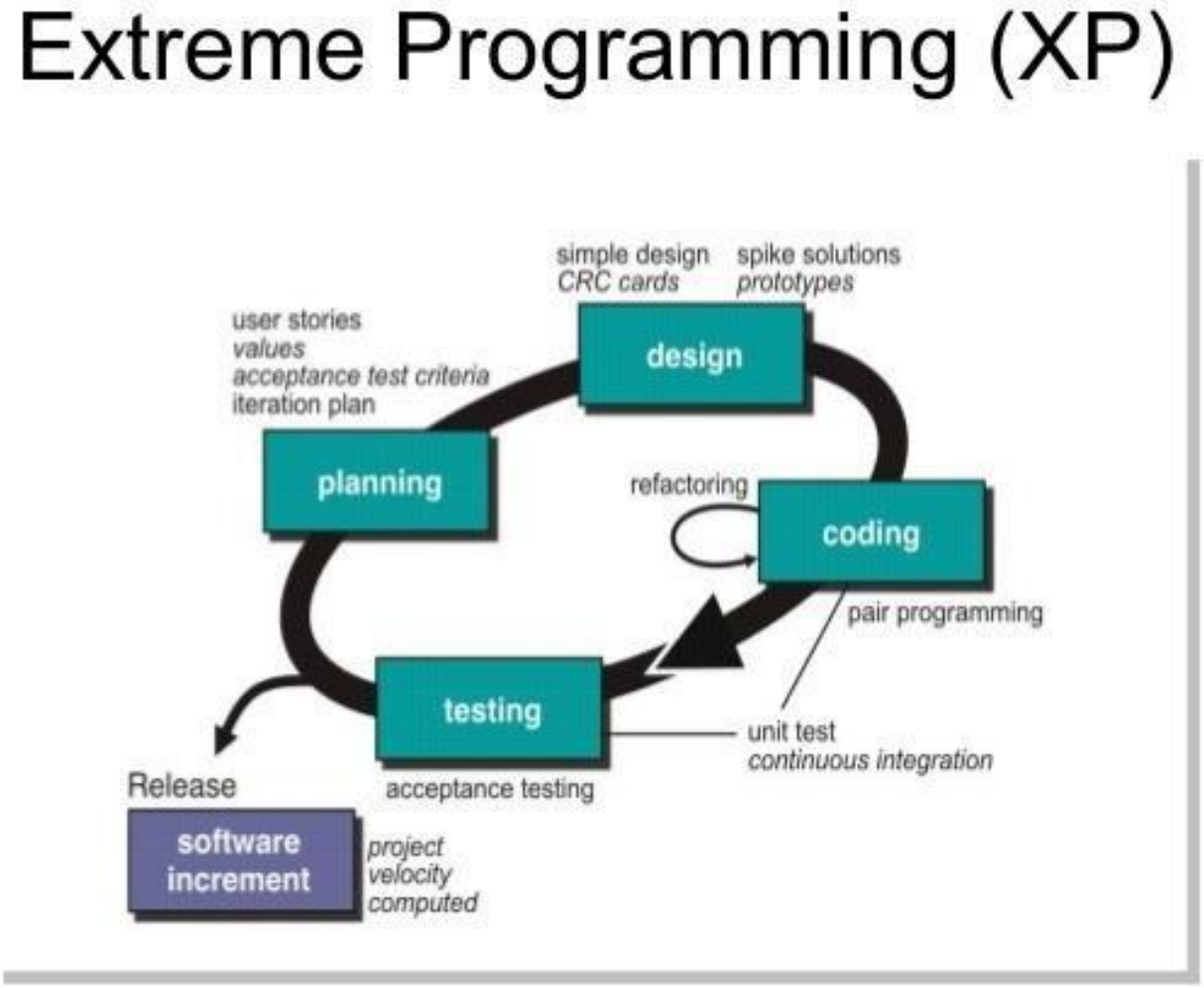


XP(Extreme Programming)

XP Values



XP(Extreme Programming)



XP(Extreme Programming)

XP Process

1.Planning

- ✓ Begins with requirement gathering
- ✓ listening leads to creation of stories that is called user stories.
- ✓ customer assign values to the story

2.Design

- ✓ xp design follows keep it simple(KIS)principle
- ✓ xp encourages use of CRC cards

3.Coding

- ✓ xp recommends two people to generate code that is called pair programming

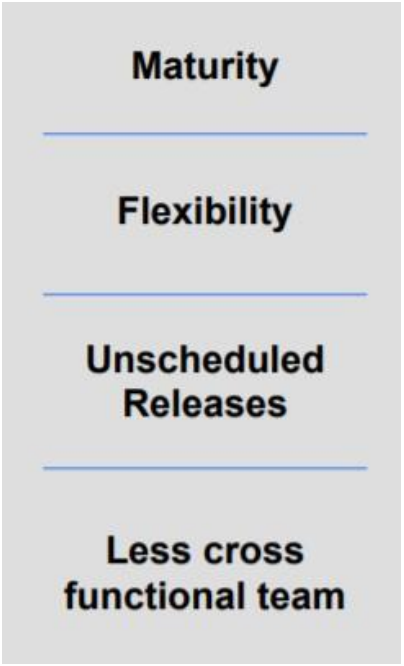
4.Testing

- ✓ unit test is done and continuous integration test is also done.
- ✓ xp acceptance test also called customer test focus on overall system feature and functionality

XP(Extreme Programming)

When to use XP

- Have reached a certain level of maturity as usual tasks, defects and smaller unrelated user stories
- Require maximum flexibility and frequent change of priorities
- Have multiple releases per week or per day
- Have many unscheduled releases
- Have less cross functional teams



XP(Extreme Programming)

Advantages

- Pair programming under XP helps in writing better codes
- Increased team accountability
- Extreme Programming manages risks in a better way
- Source code is always robust as simplicity helps in faster development and less defects
- Easy to accommodate changes

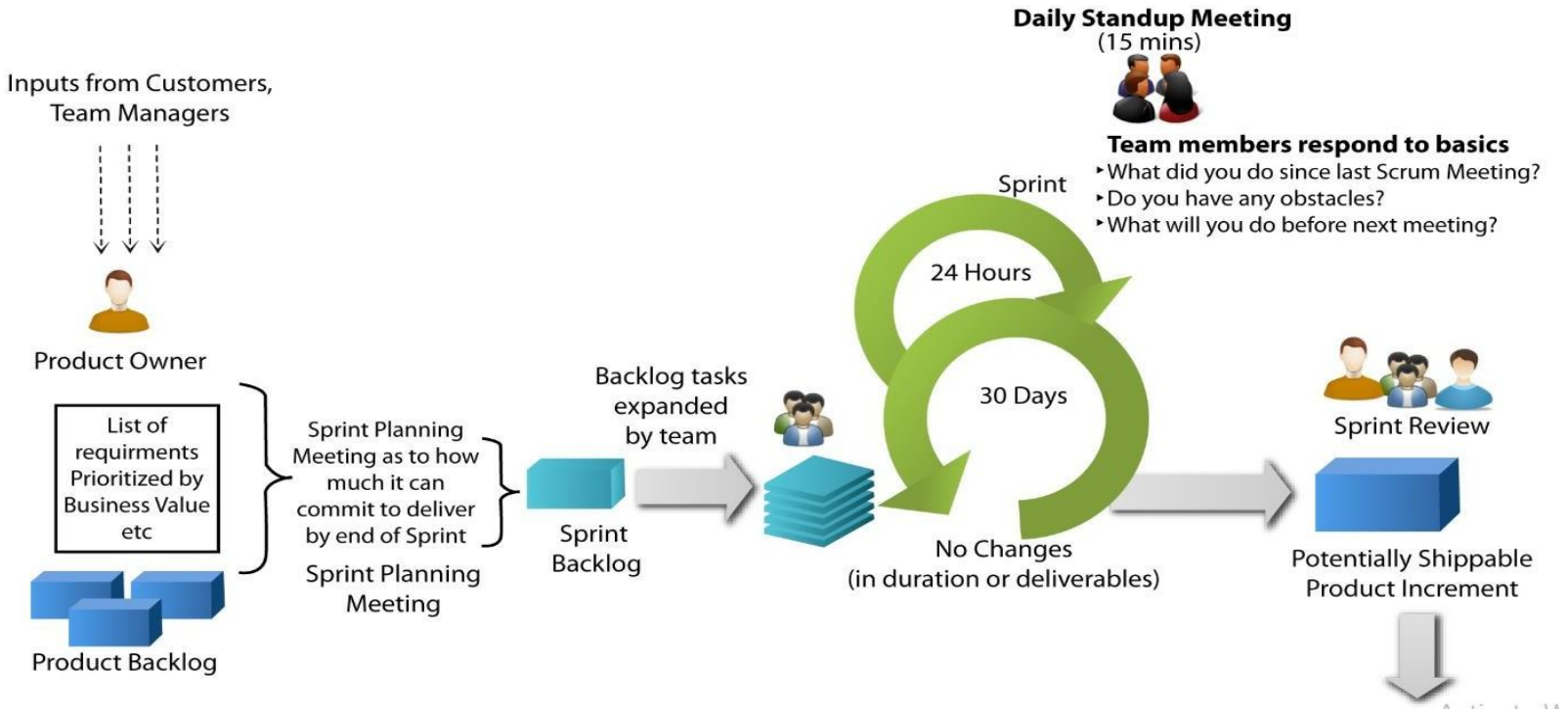
Disadvantages

- Detailed planning is required right since the inception of XP due to changing scope and associated cost
- XP does not have a set measurement plan or quality assurance for coding.
- Pair programming may lead to too much duplication of codes and data.
- XP is more code centric than design which may cause UI/UX issues in larger projects.

SCRUM

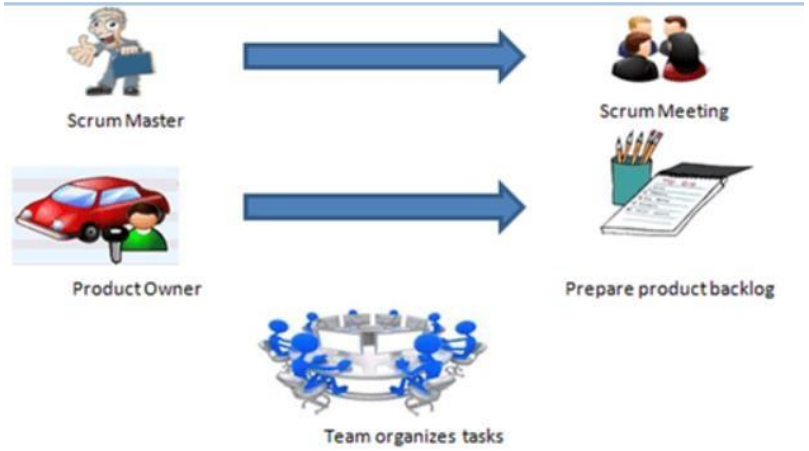
SCRUM

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment.



SCRUM

Scrum consists of three role



- **Scrum Master**

  Master is responsible for setting up the team, sprint meeting and removes obstacles to progress

- **Product owner**

  The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration

- **Scrum Team**

  Team manages its own work and organizes the work to complete the sprint or cycle

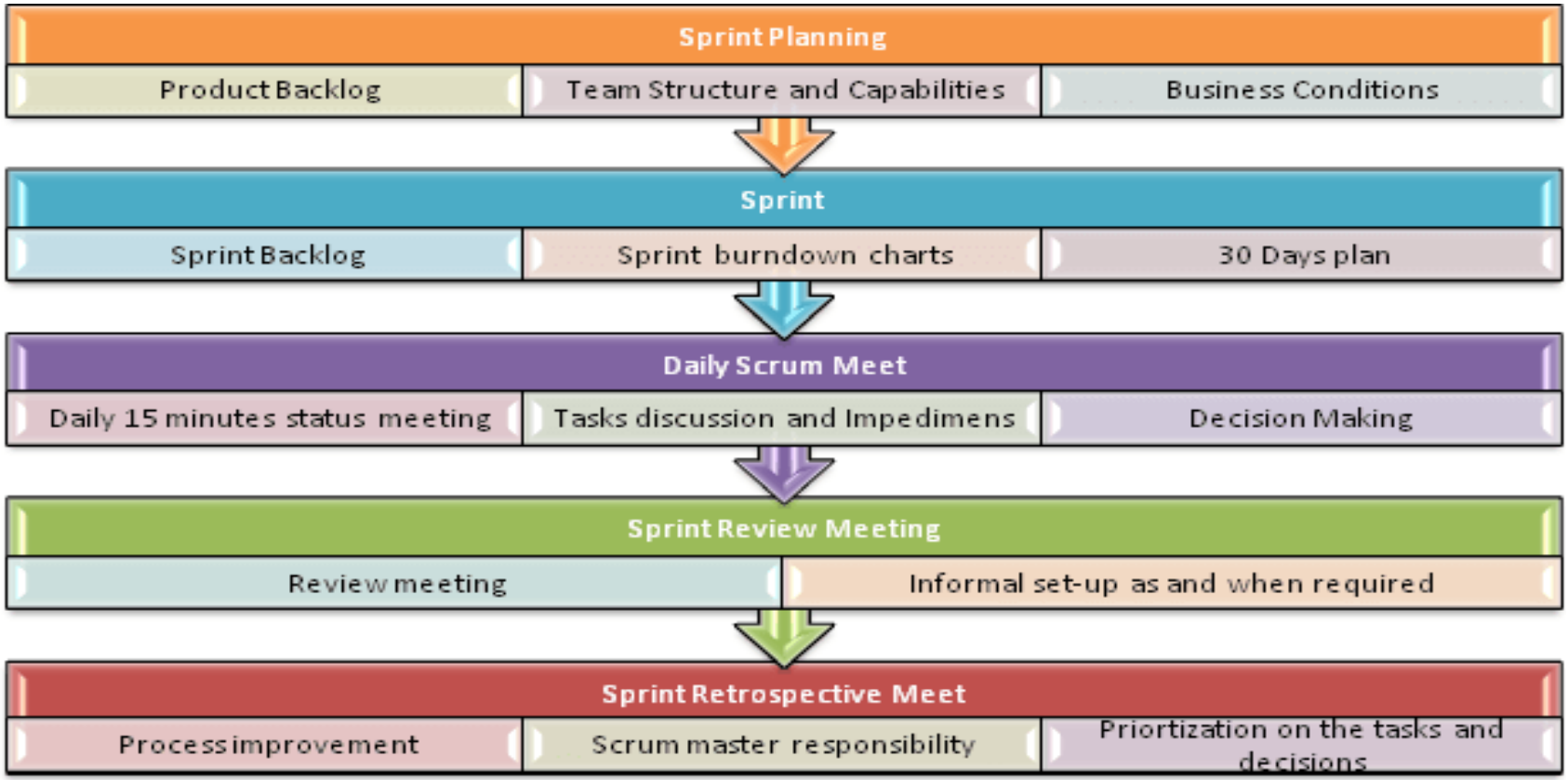
SCRUM

Scrum Process

Scrum is a hands-on system consisting of simple interlocking steps and components:

- A product owner makes a prioritized wish list known as a product backlog.
- The scrum team takes one small piece of the top of the wish list called a sprint backlog and plans to implement it.
- The team completes their sprint backlog task in a sprint (a 2-4 week period). They assess progress in a meeting called a daily scrum.
- The ScrumMaster keeps the team focused on the goal.
- At the sprint's end, the work is ready to ship or show. The team closes the sprint with a review, then starts a new sprint.

Scrum Practices



SCRUM

Who can benefit from scrum?

While scrum can benefit a wide variety of businesses and projects, these are the most likely beneficiaries

- **Complicated projects**: Scrum methodology is ideal for projects that require teams to complete a backlog.
- **Companies that value results**: Scrum is also beneficial to companies that value results over the documented progress of the process.
- **Companies that cater to customers**: Scrum can help companies that develop products in accordance with customer preferences and specifications.

SCRUM

Advantages

- Higher customer satisfaction
- Better quality
- It provides continuous feedback.
- Improved progress visibility and exposure.
- Reduce Risk
- It is easier to deliver a quality product in a scheduled time

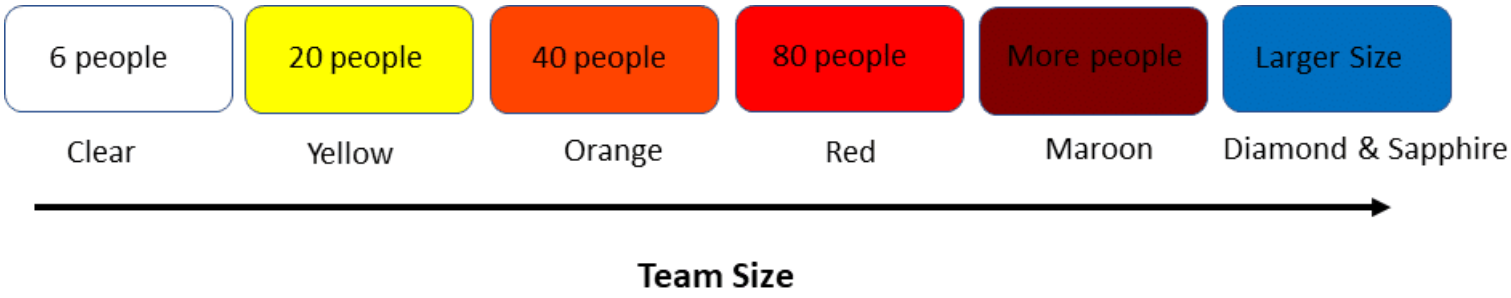
Disadvantages

- If the team members are not committed, the project will either never complete or fail.
- If a task is not well defined, estimating project costs and time will not be accurate

Crystal

- Crystal is an agile framework focusing on individuals and their interactions, as opposed to processes and tools.
- The Crystal agile framework is built on two core beliefs:

1. Teams can find ways on their own to improve and optimize their workflows
2. Every project is unique and always changing, which is why that project's team is best suited to determine how it will tackle the work



Crystal

Methods of Crystal Family

- Crystal Clear- Small team size of 1-6 people
- **Crystal Yellow**-Small team size of 7-20
- **Crystal Orange**-Team size of 21 to 40
- Crystal Orange web-Team size of 21 to 40
- Crystal Red-The traditional software development method gets followed for the team size of 40-80. In addition to that, the teams are formed and divided as the work required.
- Crystal Maroon-It is for the team size 80-200. It is for large-sized projects. Moreover, the methods defined are different & as per the need of the software.
- Crystal Diamond and Crystal Sapphire-Both are the methods used for very critical and large scale projects

Crystal

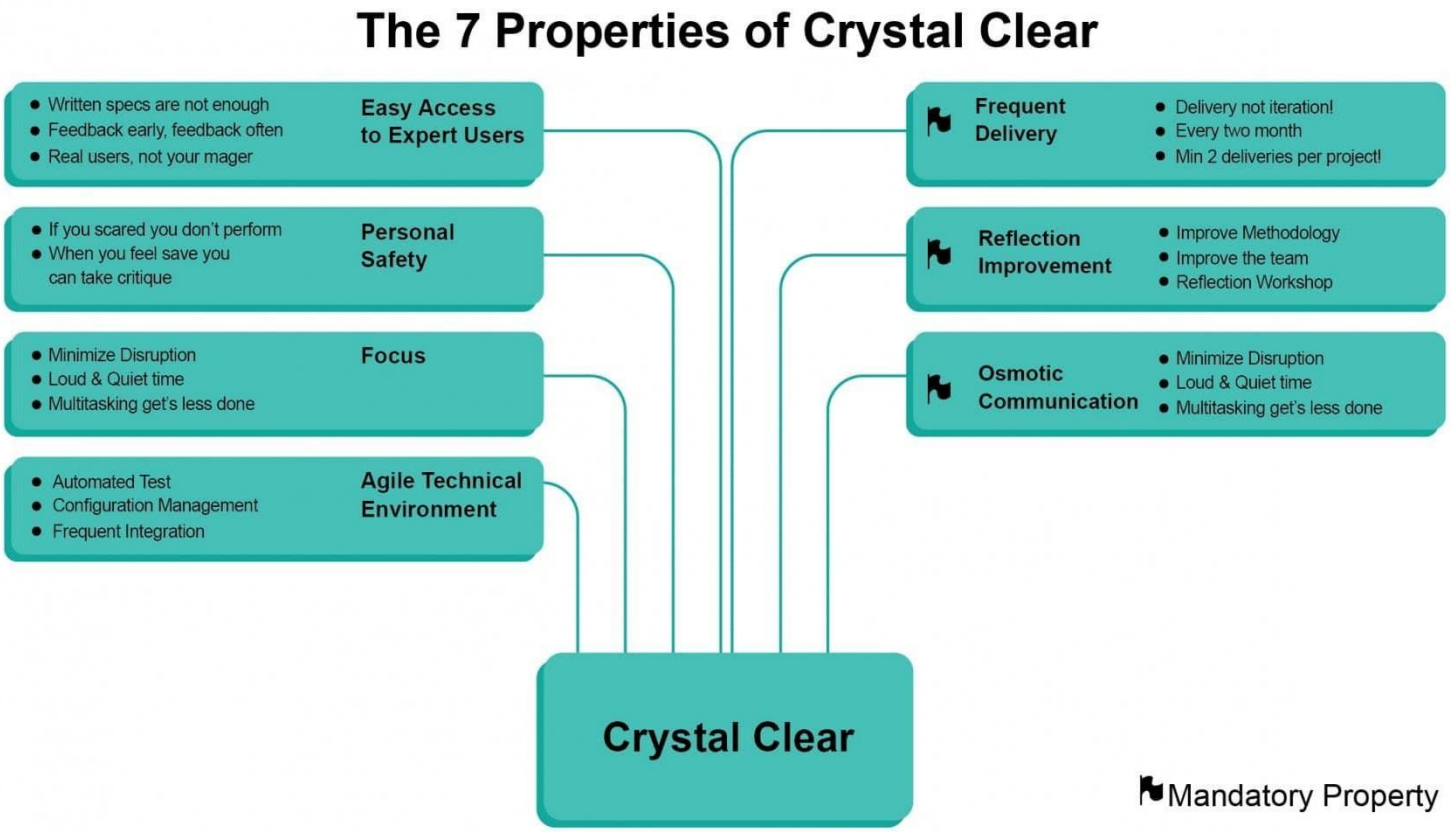
Crystal methods focus on:-

- ✓ People involved
- ✓ Interaction between the teams
- ✓ Community
- ✓ Skills of people involved
- ✓ Their Talents
- ✓ Communication between all the team

The suitability of the Crystal method depends on three dimensions:

1. Team size
2. Criticality
3. The priority of the project

Crystal


The 7 Properties of Crystal Clear
Written specs are not enough
Feedback early, feedback often
Real users, not your mager
Easy Access to Expert Users
If you scared you don't perform
When you feel save you can take critique
Personal Safety
Minimize Disruption
Loud & Quiet time
Multitasking get's less done
Focus
Automated Test
Configuration Management
Frequent Integration
Agile Technical Environment
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

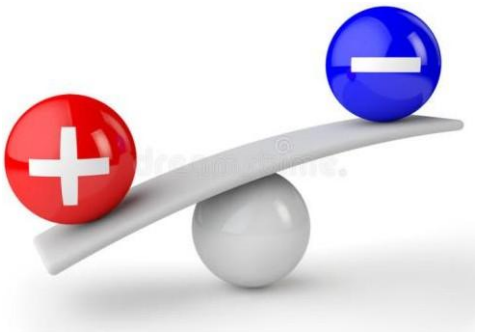
Crystal

Crystal's strengths include:

- Allows teams to work the way they deem most effective
- Facilitates direct team communication, transparency and accountability
- The adaptive approach lets teams respond well to changing requirements

Crystal's weaknesses include:

- Lack of pre-defined plans can lead to scope creep
- Lack of documentation can lead to confusion



Dynamic Systems Development Method

Dynamic Systems Development Method (DSDM)

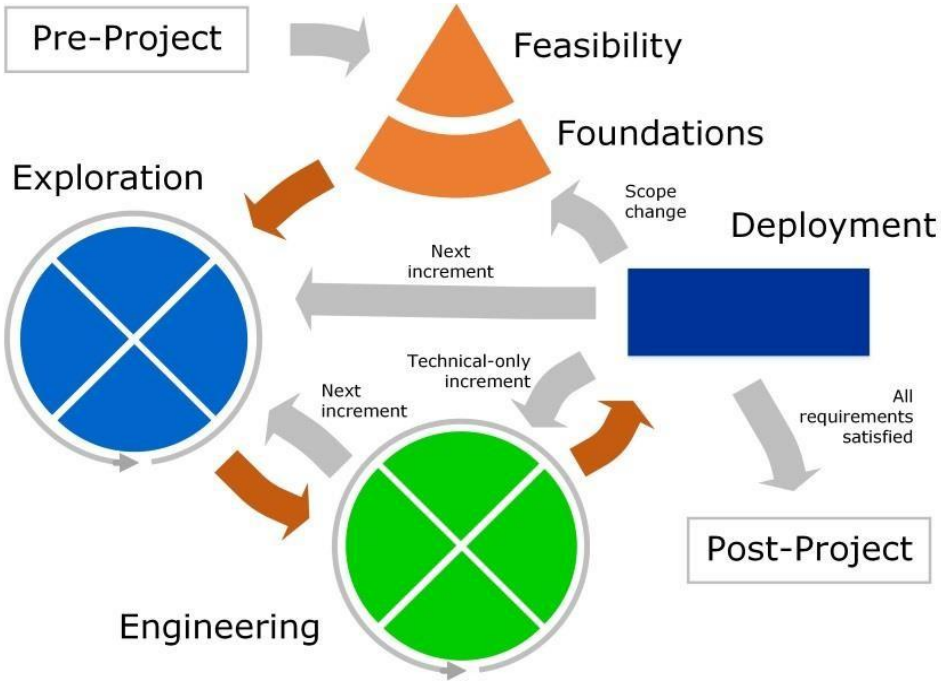
- In DSDM the users are required to be involved actively, and the **teams** are given the power to **make decisions**.
- Frequent delivery of product becomes the active focus with DSDM.
- The techniques used in DSDM are.

  1.Time Boxing
  2.MoSCoW Rules
  3.Prototyping

Dynamic Systems Development Method

The DSDM project consists of 7 phases

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project



Dynamic Systems Development Method

DSDM's strengths include:

- Basic product functionality can be delivered rapidly
- Developers have easy access to end-users
- Projects are reliably completed on time

DSDM's weaknesses include:

- Can represent a dramatic and disruptive change in company culture
- Costly to implement
- Not ideal for small organizations

Feature Driven Development (FDD)

Feature Driven Development (FDD)

- This method is focused around "**designing & building**" features.
- FDD describes very specific and short phases of work that has to be accomplished separately per feature.
- It includes domain walkthrough, design inspection, promote to build, code inspection and design.
- FDD, feature is a client value function that can be implemented in 2 weeks or less

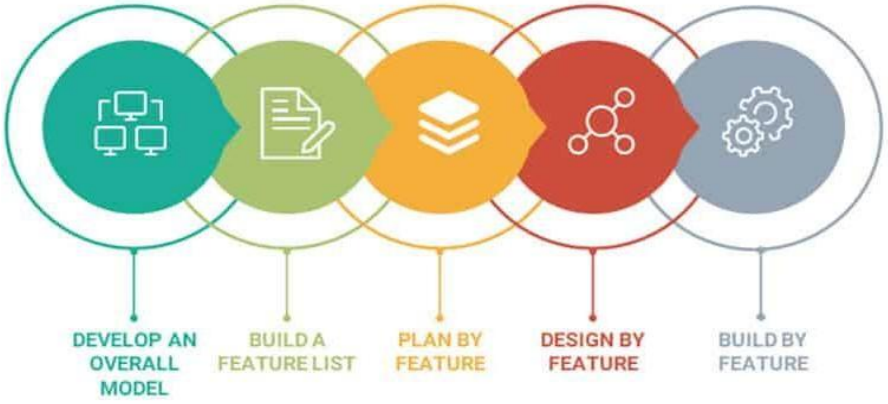
Feature Driven Development (FDD)

FDD philosophy

(1)Emphasizes collaboration among people on an FDD team.

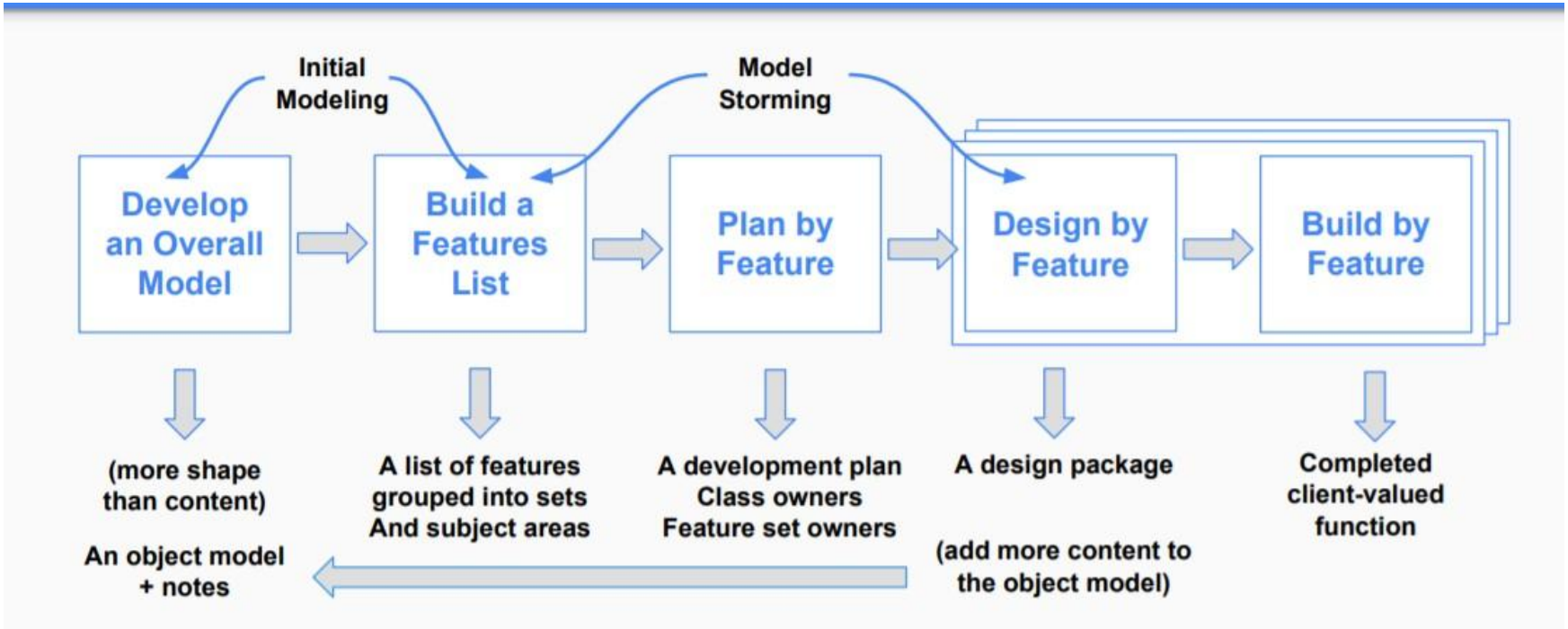
(2)Manage problem &project complexity using feature based decomposition followed by the integration of software increment

(3)Communication of technical detail using verbal graphical and text based means.



Feature Driven Development (FDD)

FDD was designed to follow a five-step development process, built largely around discrete “feature” projects.



Feature Driven Development (FDD)

FDD's strengths include:

- Simple five-step process allows for more rapid development
- Allows larger teams to move products forward with continuous success
- Leverages pre-defined development standards, so teams are able to move quickly

FDD's weaknesses include:

- Does not work efficiently for smaller projects
- Less written documentation, which can lead to confusion
- Highly dependent on lead developers or programmers

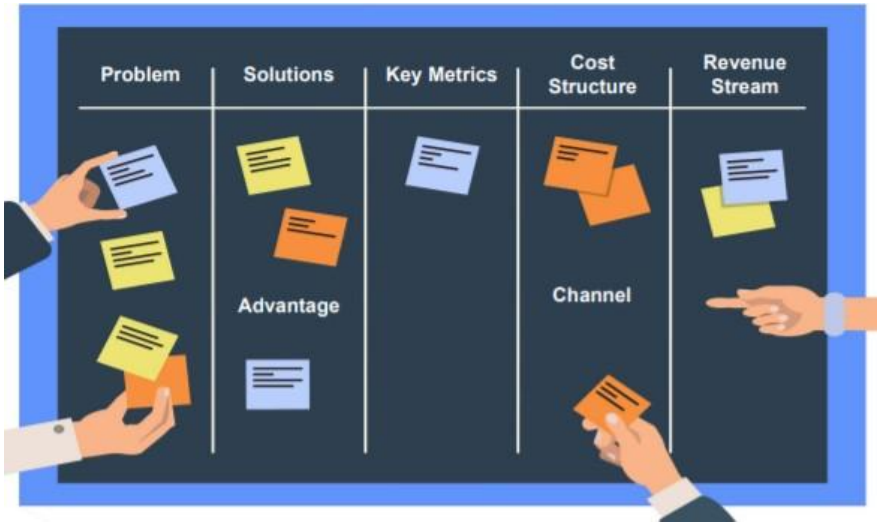
Lean software development

- **Lean software development** method is based on the principle "Just in time production".
- It aims at increasing speed of software development and decreasing cost.
- Lean development can be summarized in seven steps.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

Lean software development

- A typical lean process: Learn-Measure-Build cycle, performs quality analysis, and testing, frequently connects with clients to understand the business value and focuses on continuous improvement.
- The continuous cycle leads to sustainability, smart development, and success



Lean software development

Advantages

- Complements existing practices.
- Focuses on project ROI.
- Eliminates all project waste.

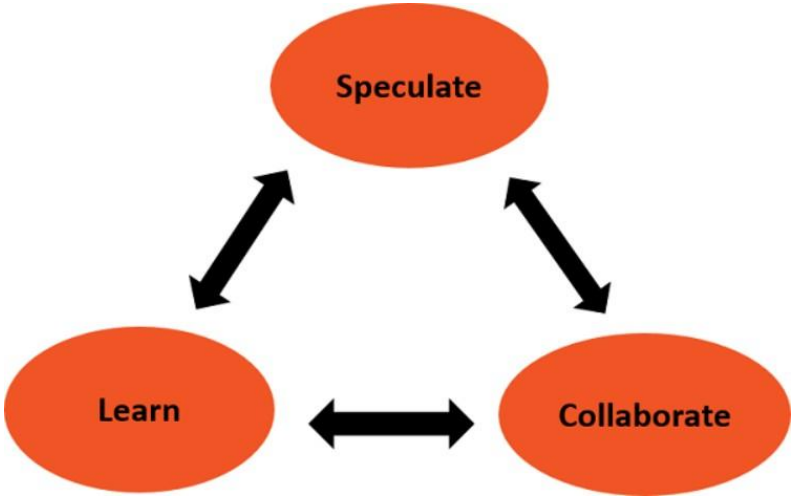
Disadvantages

- Does not specify technical practices.
- Requires constant gathering of metrics which may be difficult for some environments to accommodate.
- Theory of Constraints can be a complex and difficult aspect to adopt.

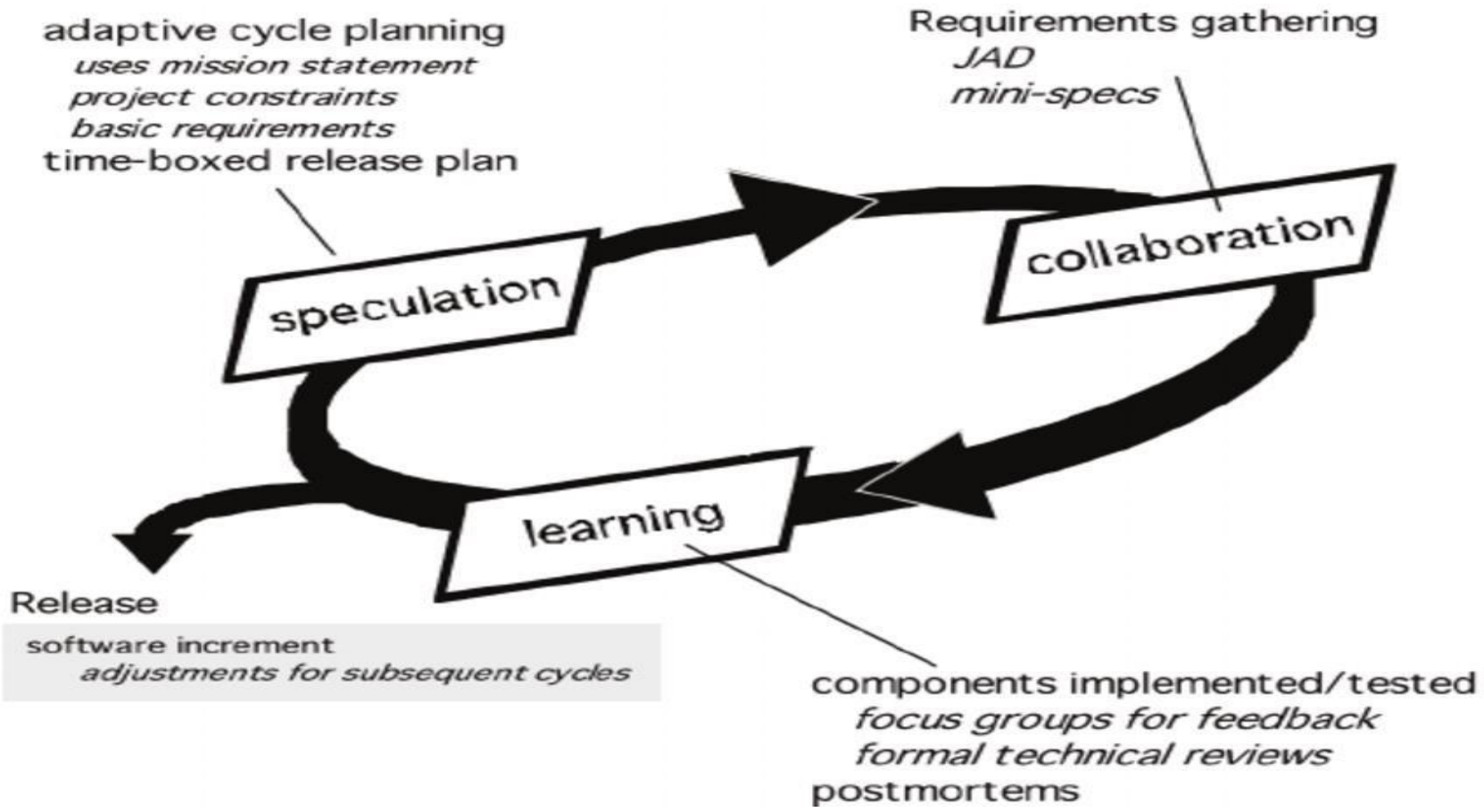
Adaptive Software Development(ASD)

- **Adaptive Software Development(ASD)** is a method to build complex software and system.
- ASD focuses on human collaboration and self-organisation. ASD **"life cycle"** incorporates three phases namely:

1. Speculation
2. Collaboration
3. Learning



Adaptive Software Development(ASD)



Adaptive Software Development(ASD)

ASD's strengths include:

- Focused on the end users, which can lead to better and more intuitive products
- Allows for on-time and even early delivery
- Encourages more transparency between developers and clients

ASD's weaknesses include:

- Demands extensive user involvement, which can be difficult to facilitate
- Integrates testing into every stage, which can add to a project's costs
- Emphasis on rapid iterating and continuous feedback can lead to scope creep

Process & Projects-Metrics

- Process
- Project
- Metric
  - ✓ what is metric?
  - ✓ who do it?
  - ✓ Why it is important?
  - ✓ why software metrics?
  - ✓ Need for software metrics
- Process metrics
- Project metrics
- Software measurement
- Metrics for software quality
- Metrics for small organization

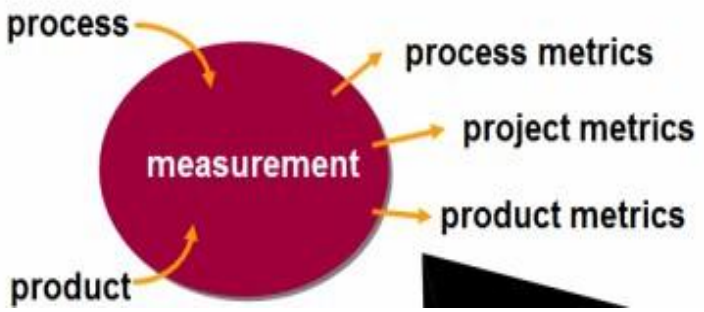
Process & Projects

Process

- Processes are established procedures for ongoing work, and they can only be changed with planning and investment. In fact, with any process that has a significant impact on a business, a project is ideally required in order to change that process.
- Processes are how work gets standardized.

Project

- A project is about creating something new or implementing a change, whereas a process is intended to create value by repeatedly performing a task
- Projects are how process work gets improved.



Process & Project-Metrics

What is metrics?

Metrics used to evaluate the performance
process and project metrics is a quantitative measures.

Who does it?

Software metrics are analyzed and assesses by software managers.

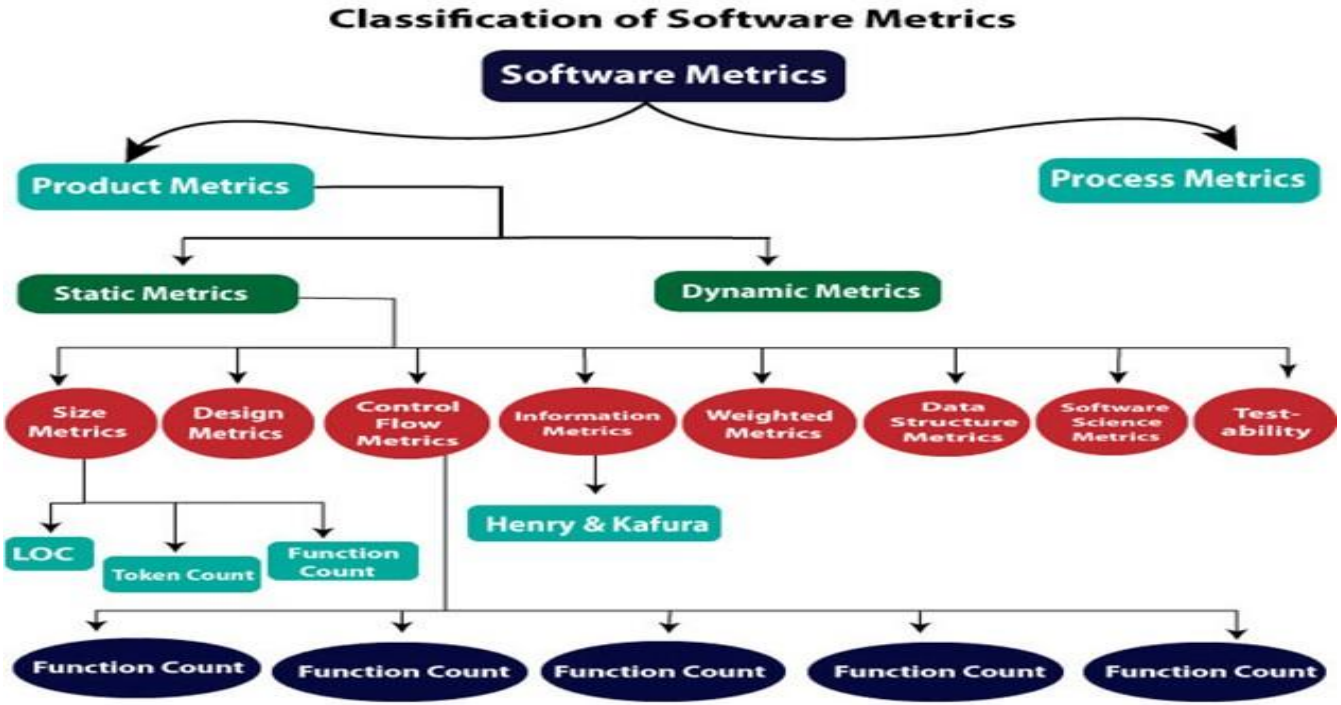
Why is it important?

- If you don't measure, judgment can be based only on subjective evaluation.

  With measurement, trends (either good or bad) can be spotted, better estimates can be made, and true improvement can be accomplished over time.

Process & Project-Metrics

- A **software metric** is a measure of software characteristics which are measurable or countable.
- Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.



Process & Project-Metrics

Why software metrics???

- To characterize.
- To evaluate.
- To predict.
- To improve.

Process & Project-Metrics

- **Need for Software Metrics.**

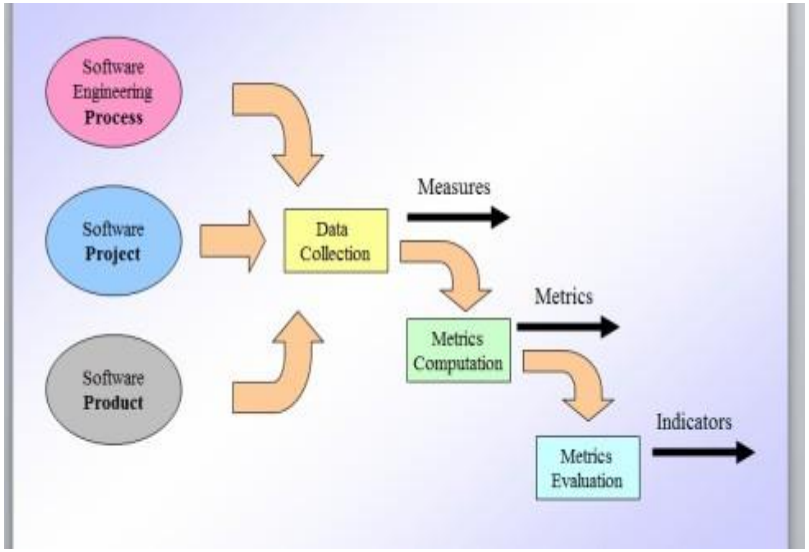
Following are the needs for the software Metrics

- To **characterize** in order to
  - ✓ Gain an understanding of processes, products, resources, and environments.
  - ✓ Establish baselines for comparisons with future assessments
- To **evaluate** in order to determine status with respect to plans
- To **predict** in order to
  - ✓ Gain understanding of relationships among processes and products.
  - ✓ Build models of these relationships
- To **improve** in order to
  - ✓ Identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance

Process & Project-Metrics

Process Metrics

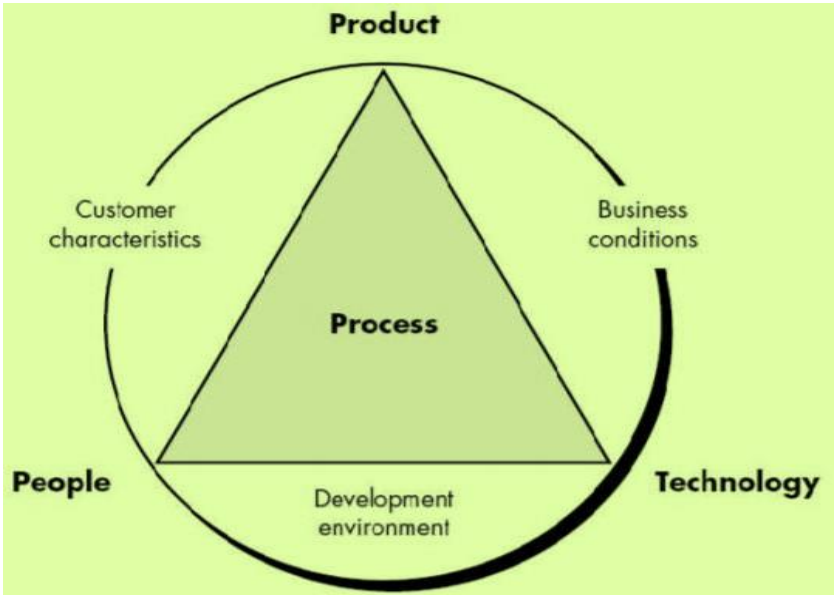
- **Process metrics** can be used to improve software development and maintenance.
- **Examples**
  - ✓ include the effectiveness of defect removal during development,
  - ✓ the pattern of testing defect arrival,
  - ✓ and the response time of the fix process.



Process & Project-Metrics

Project metrics

- **Project metrics** describe the project characteristics and execution.
- Examples
  - ✓ include the number of software developers,
  - ✓ the staffing pattern over the life cycle of the software,
  - ✓ cost and Schedule
  - ✓ productivity.



Process & Projects-Metrics

Software measurement

- Measure-provides a quantitative indication of the size of some product or process.
- Measurement-is the act of obtaining a measure.

Types of software measurement

- **Direct Measurement-** the product, process or thing is measured directly using standard scale.
- **Indirect Measurement-** the quantity or quality to be measured is measured using related parameter i.e. by use of reference.

Normally measurement is useful for,

- Understanding the process and products
- Establishing a baseline
- Accessing and predicting the outcome

Process & Projects-Metrics

1.Size-Oriented Metrics

- Size-oriented software metrics are derived by **normalizing quality and/or productivity** measures by considering the size of the software that has been produced.
- A set of simple size-oriented metrics can be developed for each project:
  - ✓ Errors per KLOC (thousand lines of code)
  - ✓ Defects per KLOC
  - ✓ $ per KLOC

In addition, other interesting metrics can be computed:

- ✓ Errors per person-month
- ✓ KLOC per person-month
- ✓ $ per page of documentation

Process & Projects-Metrics

2.Function-Oriented Metrics

- It use a measure of the functionality delivered by the application as a normalization value.
- The most widely used function-oriented metric is the function point (FP).
- Computation of the function point is based on characteristics of the software's information domain and complexity.

3.Reconciling LOC and FP Metrics

- The relationship between LOC & FP depends upon the programming language that is used to implement the software and the quality of the design
- FP & LOC-based metrics are relatively accurate
- predictors of software development effort and cost.

4.Object-Oriented Metrics

- Lines of code and functional point metrics can be used for estimating object-oriented software projects.
- However, these metrics are not appropriate in the case of incremental software development as they do not provide adequate details for effort and schedule estimation.
- Thus, for object-oriented projects, different sets of metrics have been proposed. These are listed below.
  - ✓ Number of key classes:
  - ✓ Number of support classes:
  - ✓ Number of scenario scripts:
  - ✓ Average number of support classes per key class:
  - ✓ Number of subsystems:

Process & Projects-Metrics

5.Use-Case–Oriented Metrics

- Use cases are used widely as a method for **describing customer-level or business domain requirements** that imply software features and functions.
- use-case points (UCPs) as a mechanism for estimating project effort and other characteristics.

6.WebApp Project Metrics

- The objective of all WebApp projects is to **deliver a combination of content and functionality to the end user.**
- Measures that can be collected are:

1.Number of static Web page
2.Number of dynamic Web pages.
3.Number of internal page links
4.Number of persistent data objects
5.Number of external systems interfaced
6.Number of static content objects
7.Number of dynamic content objects
8.Number of executable functions

Software Quality Metrics

- Although there are many measures of software quality, correctness, maintainability, integrity, and usability provide useful indicators for the project team.
- To measure integrity, two additional attributes must be defined: threat and security.
  - ✓ **Threat** is the probability (which can be estimated or derived from empirical evidence) that an attack of a specific type will occur within a given time.
  - ✓ **Security** is the probability (which can be estimated or derived from empirical evidence) that the attack of a specific type will be repelled

Process & Projects-Metrics

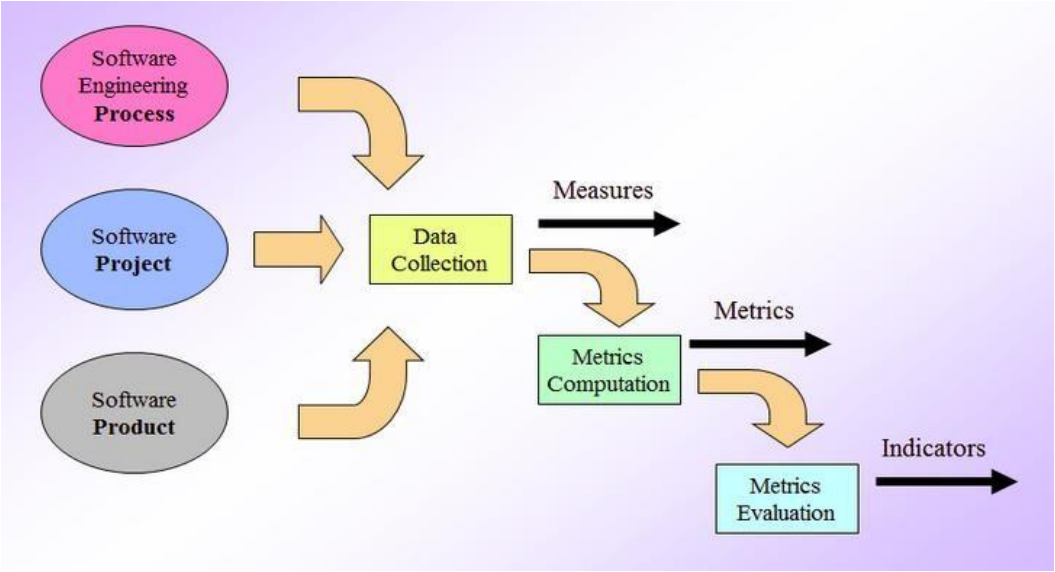
- **Defect removal efficiency (DRE)** is a measure of the filtering ability of the quality assurance and control activities as they are applied through out the process framework

  DRE = E / (E + D)

  E = number of errors found before delivery of work product

  D = number of defects found after work product delivery

- **Integrating metrics with software process**



Process & Projects-Metrics

Metrics for small organization

A small organization might select the following set of easily collected measures:

- ✓ Time (hours or days) elapsed from the time a request is made until evaluation is complete
- ✓ Effort (person-hours) to perform the evaluation.
- ✓ Time (hours or days) elapsed from completion of evaluation to assignment of change order to personnel.
- ✓ Effort (person-hours) required to make the change.
- ✓ Time required (hours or days) to make the change.
- ✓ Errors uncovered during work to make change.
- ✓ Defects uncovered after change is released to the customer base.

Project Management

- What is a project
- What is software project management?
- Need for software project management
- What are the stages of project management?
- Why is project management important?
- Project manager roles and responsibilities
- Goals of project management
- Activities

Project Management

What is a project

- A project is a group of tasks that need to complete to reach a clear result.
- A project also defines as a set of inputs and outputs which are required to achieve a goal.
- Projects can vary from simple to difficult and can be operated by one person or a hundred.

Project Management

What is Software project management?

- Software project management is an art and discipline of planning and supervising software projects.
- It is a sub-discipline of software project management in which software projects planned, implemented, monitored and controlled.
- It is a procedure of managing, allocating and timing computer software that fulfills requirements.
- In software Project Management, the client and the developers need to know the length, period and cost of the project.

Project Management

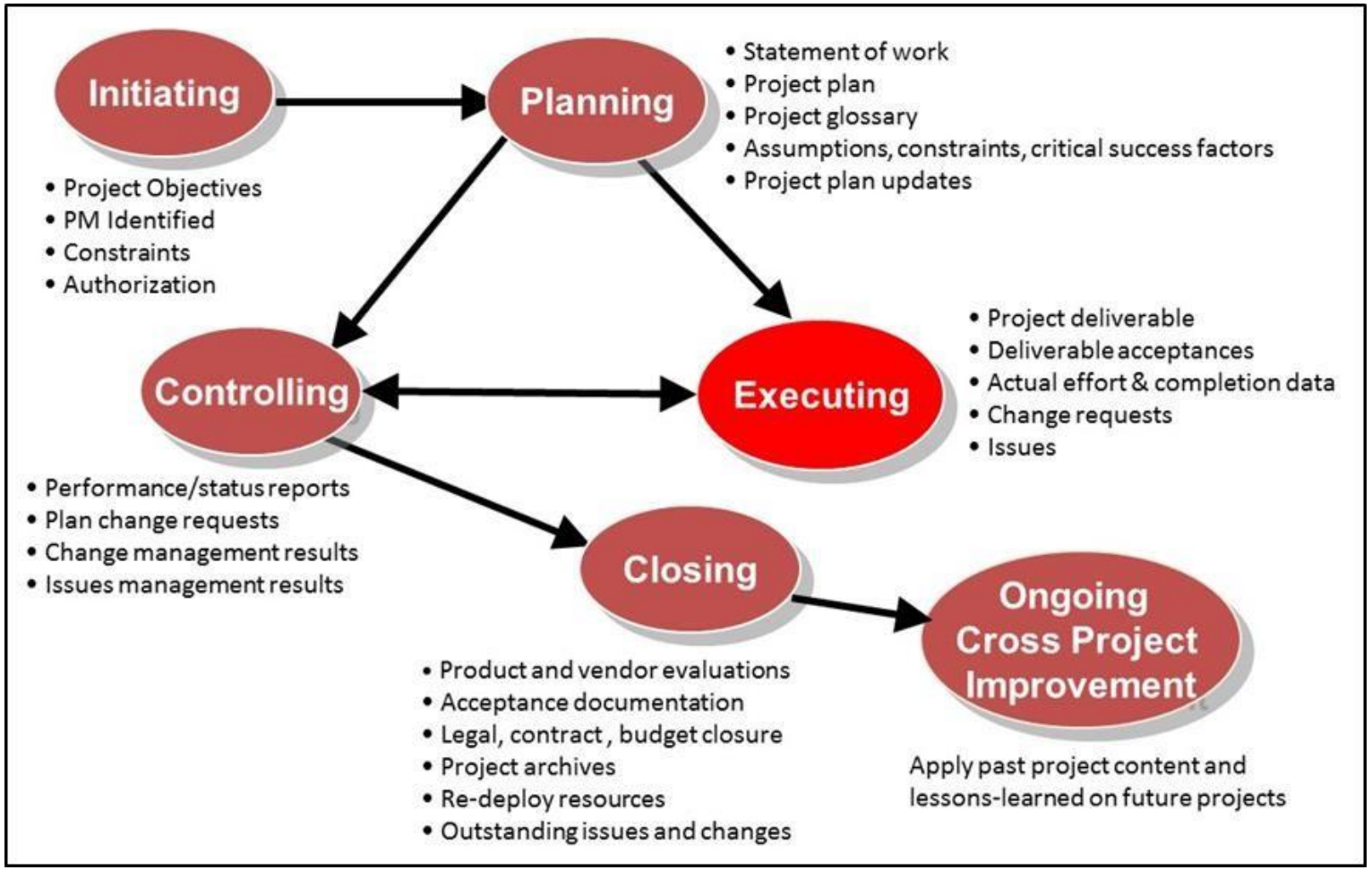
Need for software project management

There are three needs for software project management.

- ✓ Time
- ✓ Cost
- ✓ Quality

Project Management

What are the stages of project management?



Project Management

1.Initiating: define the project.

2.Planning: developing a roadmap for everyone to follow.

3.Executing & Monitoring: the project team is built and deliverables are created. Project managers will monitor and measure project performance to ensure it stays on track.

4.Closing: The project is completed, a post mortem is held, and the project is transferred to another team who will maintain it.

Project Management

Why is project management important?

Project managers will help your organization:

- have a more predictable project planning and execution process
- adhere to project budgets, schedules, and scope guidelines
- resolve project roadblocks and escalate issues quicker and easier
- identify and terminate projects that do not have relevant business value
- become more efficient
- improve collaboration across and within teams
- identify and plan for risks

Project Management

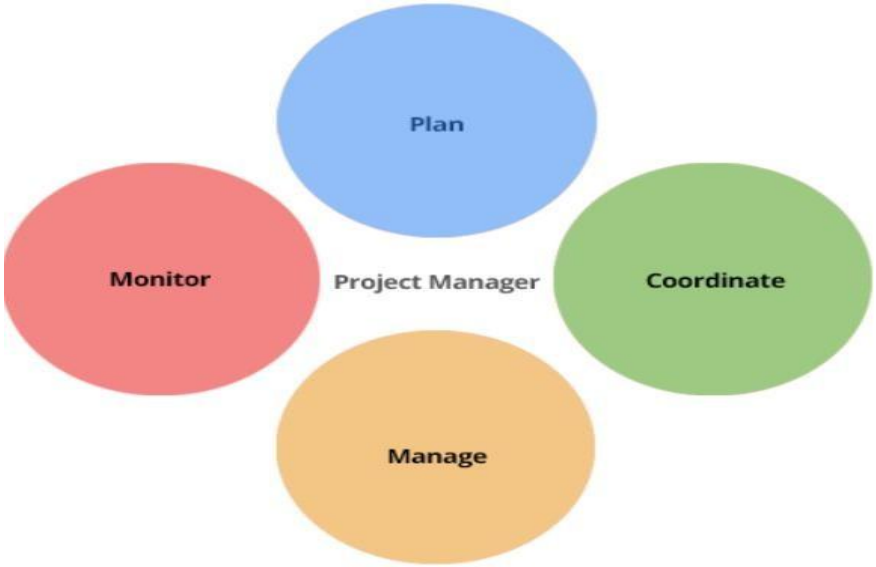
Project manager

- A project manager is a character who has the overall responsibility for the planning, design, execution, monitoring, controlling and closure of a project.

Project Management

Roles and Responsibility of a Project Manager

- Managing risks and issues.
- Create the project team and assigns tasks to several team members.
- Activity planning and sequencing.
- Monitoring and reporting progress.
- Modifies the project plan to deal with the situation.



Project Management

Goals of Project management

Project Management

Activities

Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc

- ✓ Project planning and Tracking
- ✓ Project Resource Management
- ✓ Scope Management
- ✓ Estimation Management
- ✓ Project Risk Management
- ✓ Scheduling Management
- ✓ Project Communication Management
- ✓ Configuration Management

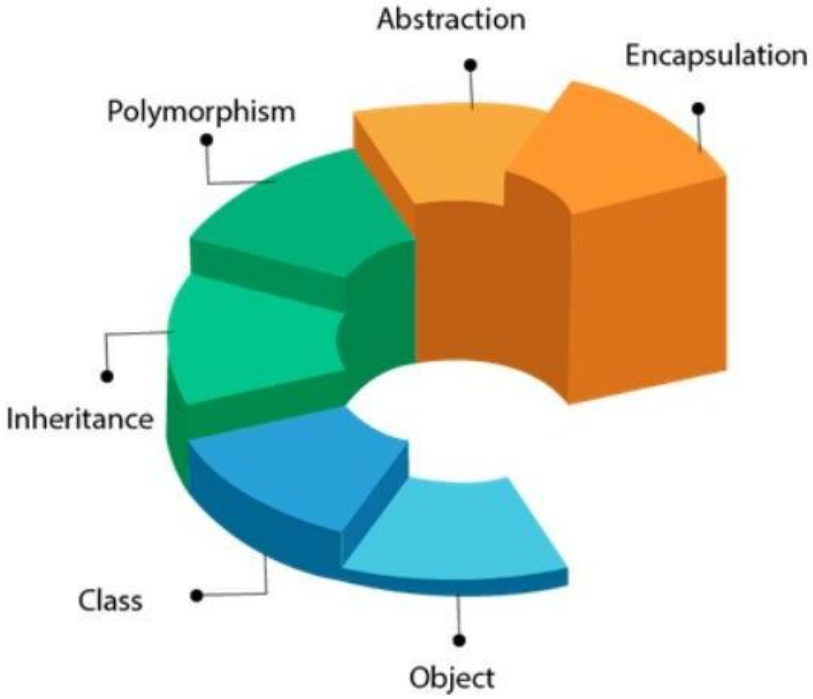
OO concepts, principles & Methodology

OO Concepts & Principles

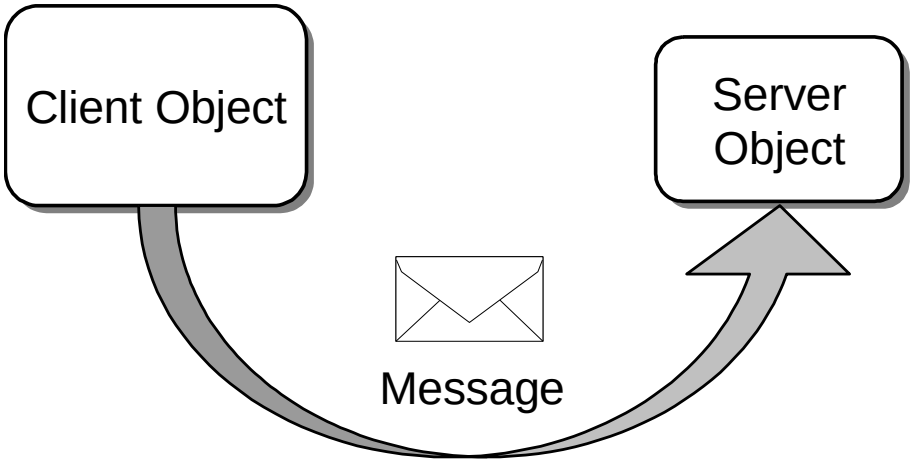
- ✓ Object & class
- ✓ Message & method
- ✓ Encapsulation
- ✓ Inheritance
- ✓ Polymorphism
- ✓ Abstraction

OO Methodology

- ✓ Booch Methodology
- ✓ Rumbaugh Methodology
- ✓ Jacobson Methodology
- ✓ Rational Unified Process(RUP)



OO concepts, principles & Methodology

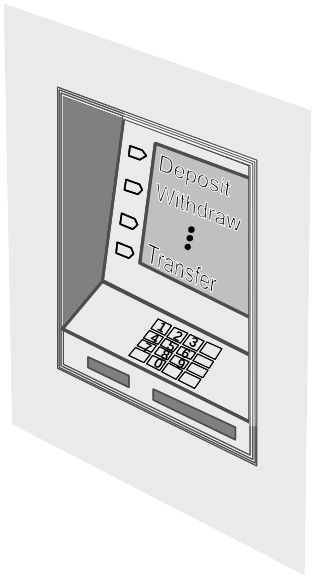


- Objects send **messages** by calling methods
- **Client object**: sends message and asks for service
- **Server object**: provides service" and returns result

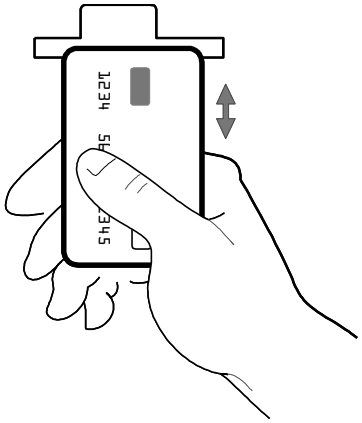
Acceptable calls are defined by object "**methods**"

(a.k.a. Operations, Procedures, Subroutines, Functions)

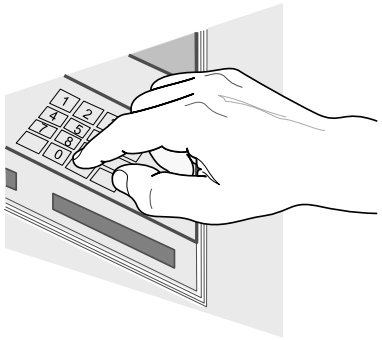
Object:
ATM machine



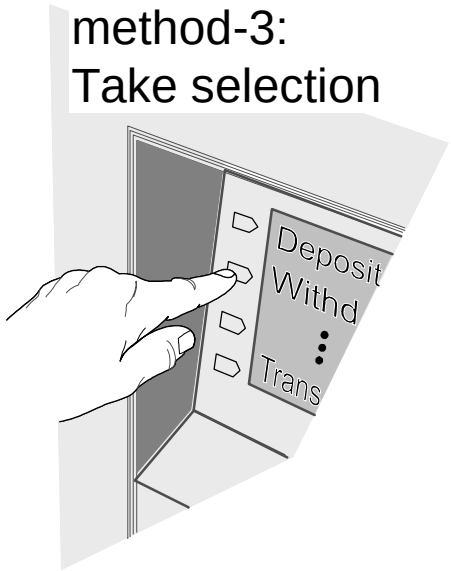
method-1:
Accept card



method-2:
Read code



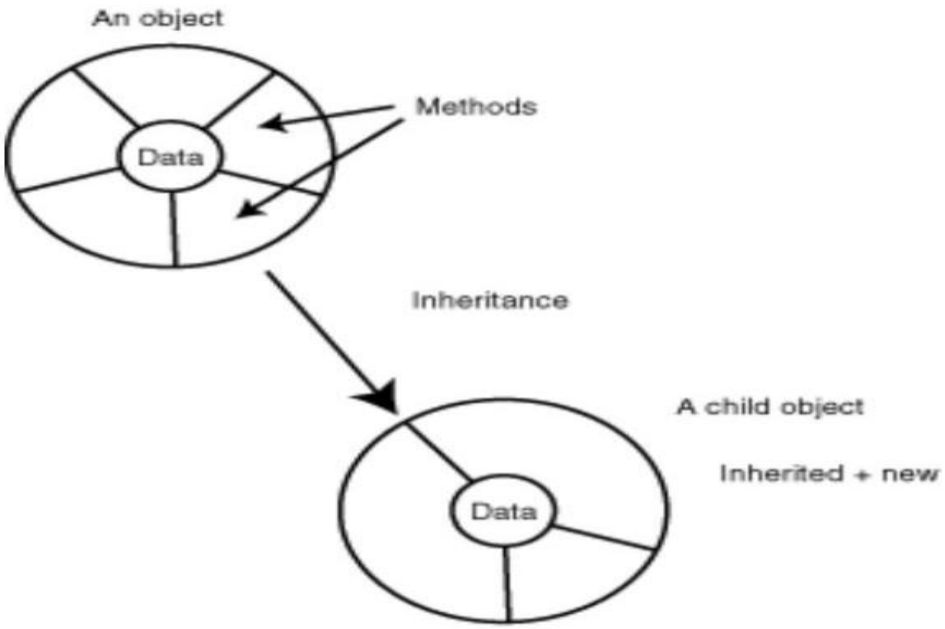
method-3:
Take selection



OO concepts, principles

It is a popular design for analyzing and designing an application to improve the software development and maintainability.

What is object orientation



OO concepts, principles

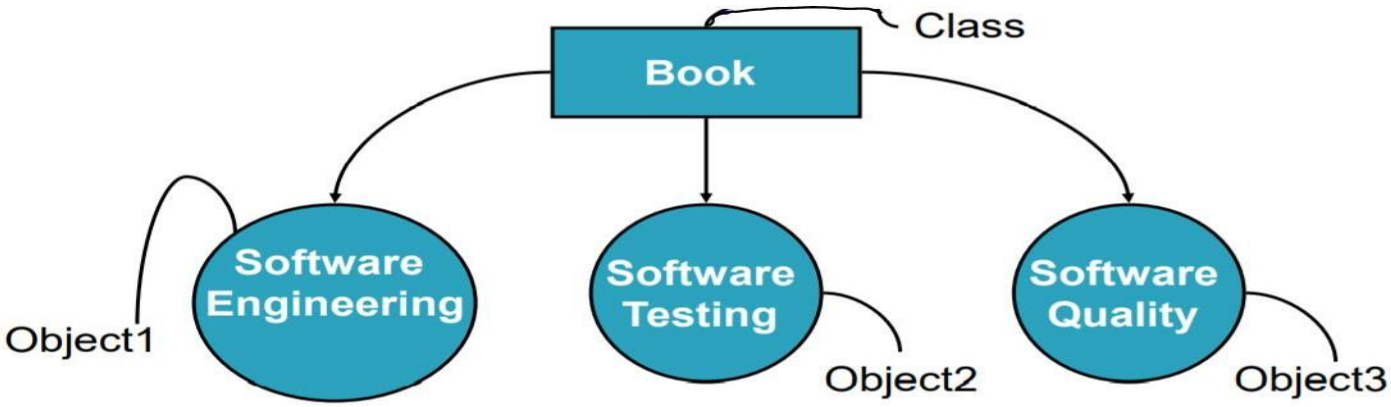
Classes & Objects

Object

- Any entity that has **state and behavior** is known as an object.
- For example: chair, pen, table, keyboard, bike etc.

Class

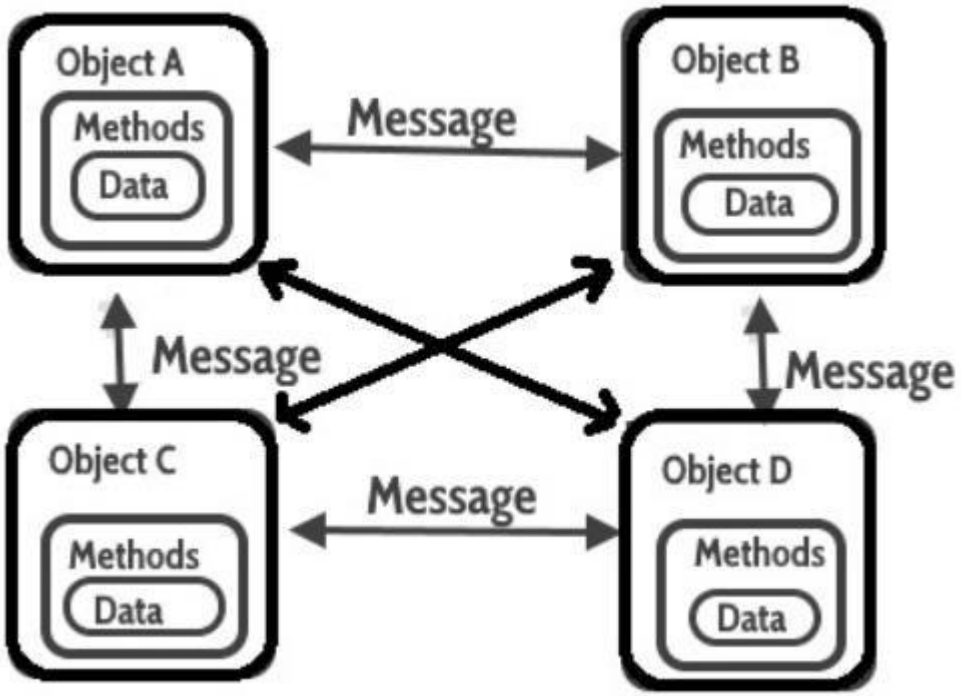
- A class is simply a representation of a type of object.
- It is the **blueprint, or plan, or template**, that describes the details of an object



OO concepts, principles

Messages

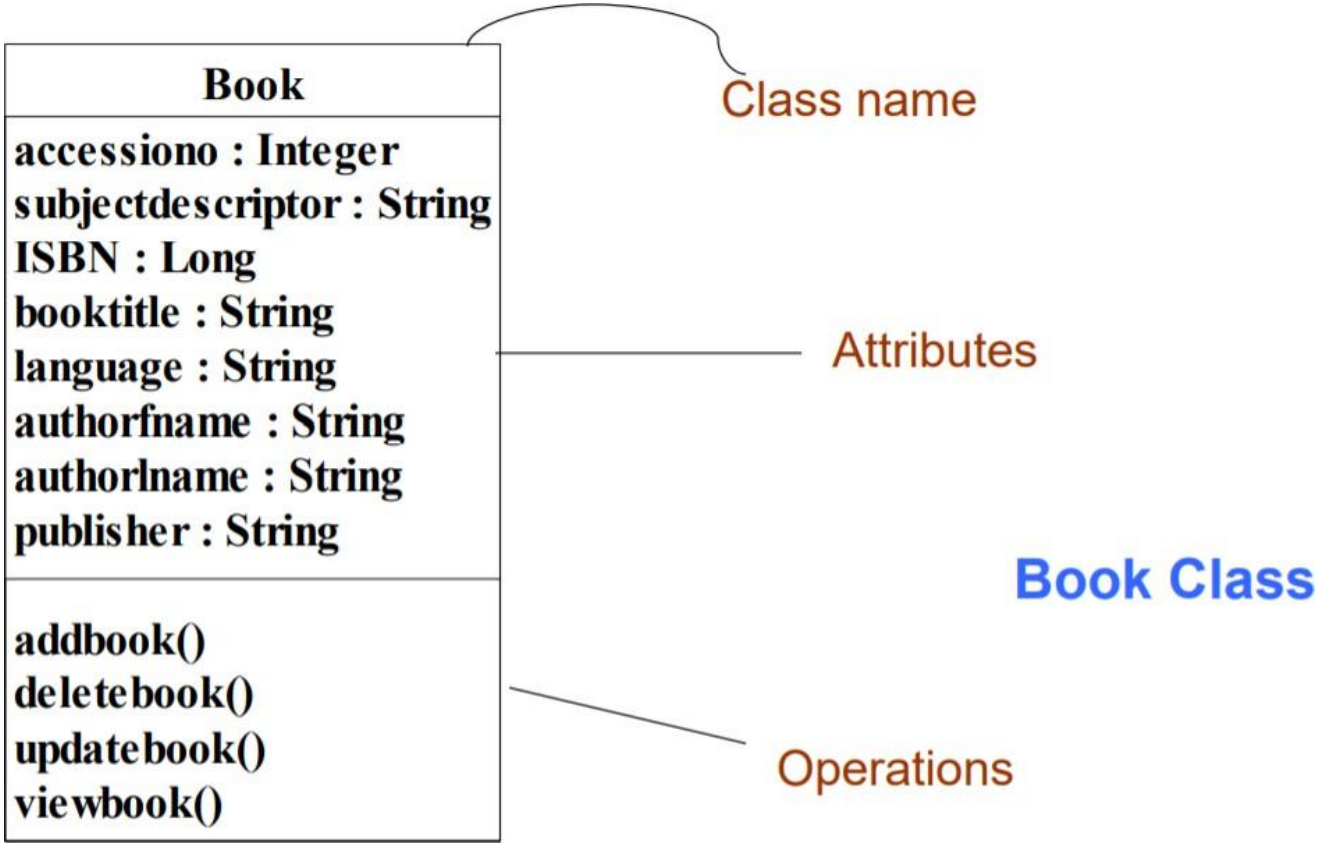
- Objects c**ommunicate** through passing messages.
- An object which originates a message is called the **sender** and the object which receives a message is called the **receiver.**



OO concepts, principles

Method

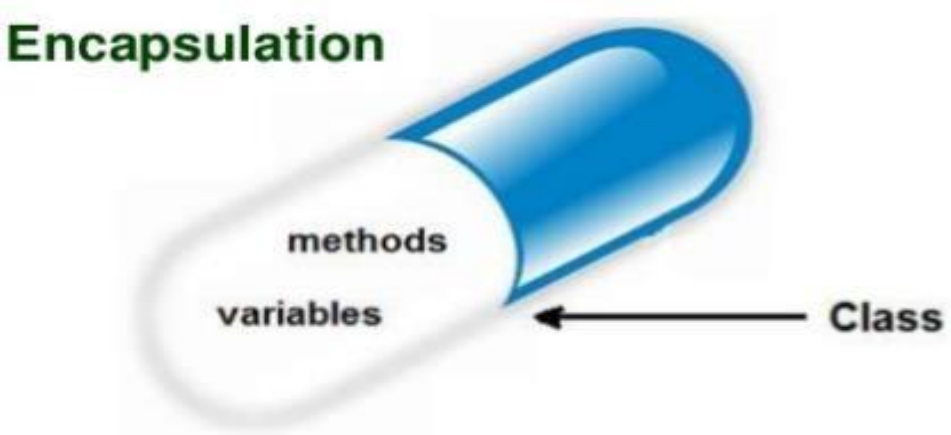
➢A 'method' is the sequence of steps (or set of operations) to be performed to fulfill the assigned task.



OO concepts, principles

Encapsulation

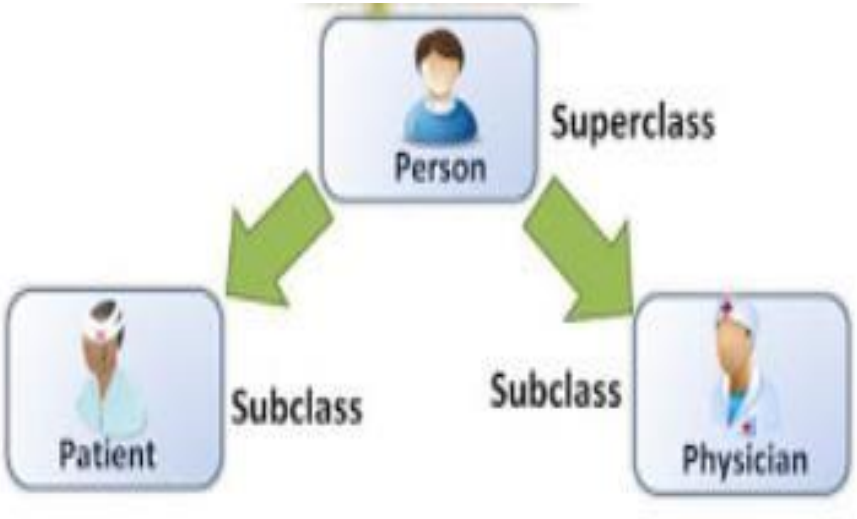
- The wrapping up of data and functions into a single unit is known as encapsulation.
- The data is not accessible to the outside world, only those function which are wrapped in the can access it.
- These functions provide the interface between the object's data and the program.
- This ensulation of the data from direct access by the program is called data hiding or information hiding



OO concepts, principles

Inheritance

- When one object acquires all the properties and behaviors of a parent object, it is known as inheritance.
- The parent class is called the **base class** or **super class**. The child class that extends the base class is called the derived class or **sub class** or **child class**.



OO concepts, principles

Polymorphism

- Poly means **Many forms.**
- If **one task is performed in different ways**, it is known as polymorphism.
- **Example:**
  - ✓ to speak something; for example, a cat speaks meow, dog barks woof, etc.

Data Abstraction

- Hiding internal details and showing functionality is known as abstraction.
- **Example:**

Abstraction shows only important things to the user and hides the internal details, for example, when we ride a bike, we only know about how to ride bikes but can not know about how it work? And also we do not know the internal functionality of a bike.

OO concepts, principles

Coupling-refers to the knowledge or information or dependency of another class

Cohesion-refers to the level of a component which performs a single well-defined task

Association-represents the relationship between objects

1-1

1-M

M-1

M-M

Aggregation-has a

Composition

Object Oriented Methodologies

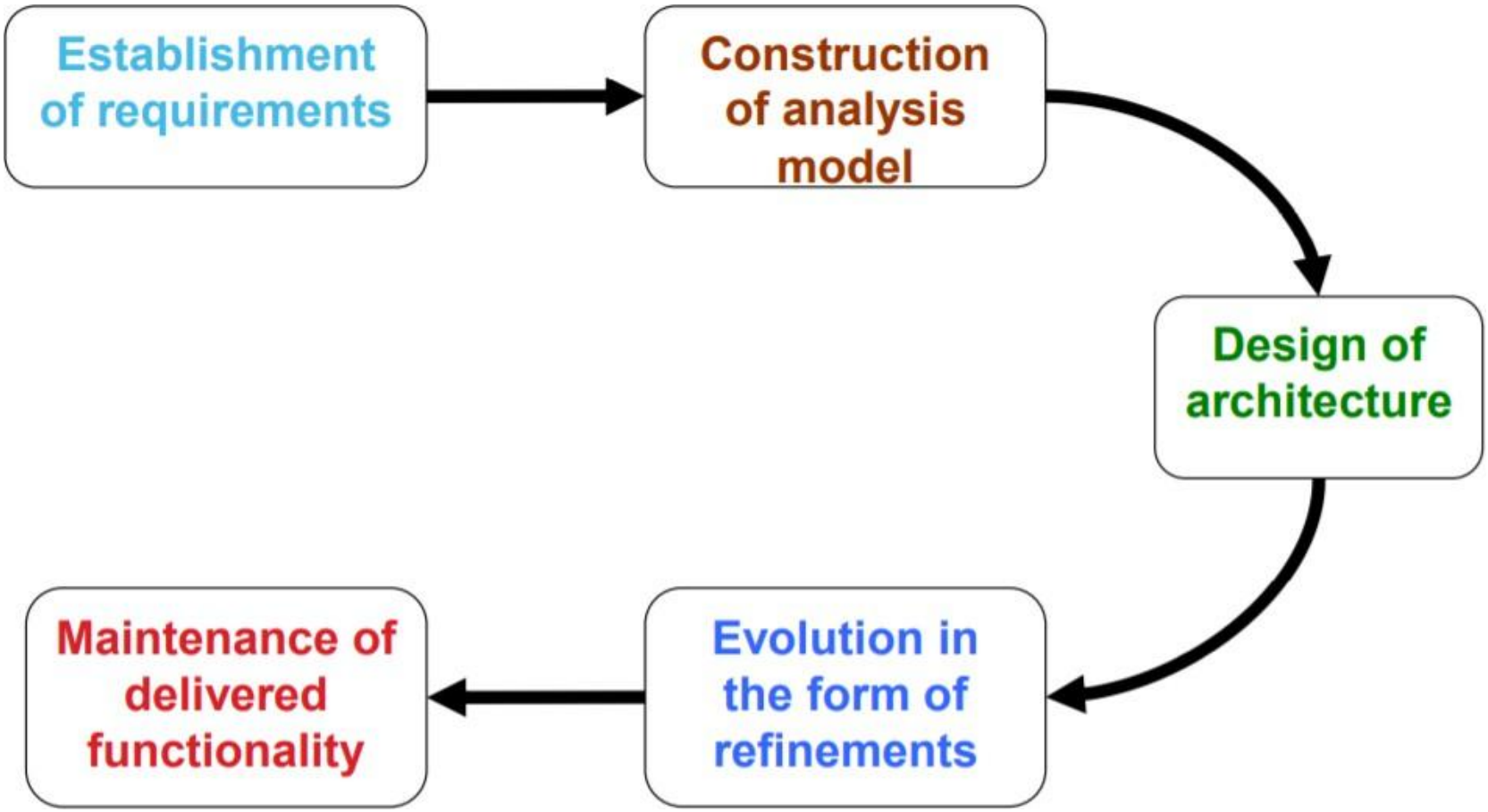
OO Methodologies

- It is a new system development approach encouraging and facilitating re-use of software component

Booch Methodology

- Grady Booch proposed object oriented methodology in his book Object-Oriented Design (OOD) in 1991.
- The primary aim of OOD was to establish a base for implementation of object oriented systems.
- The Booch methodology can be broadly divided into two processes: macro process and micro process

Object Oriented Methodologies



Object Oriented Methodologies

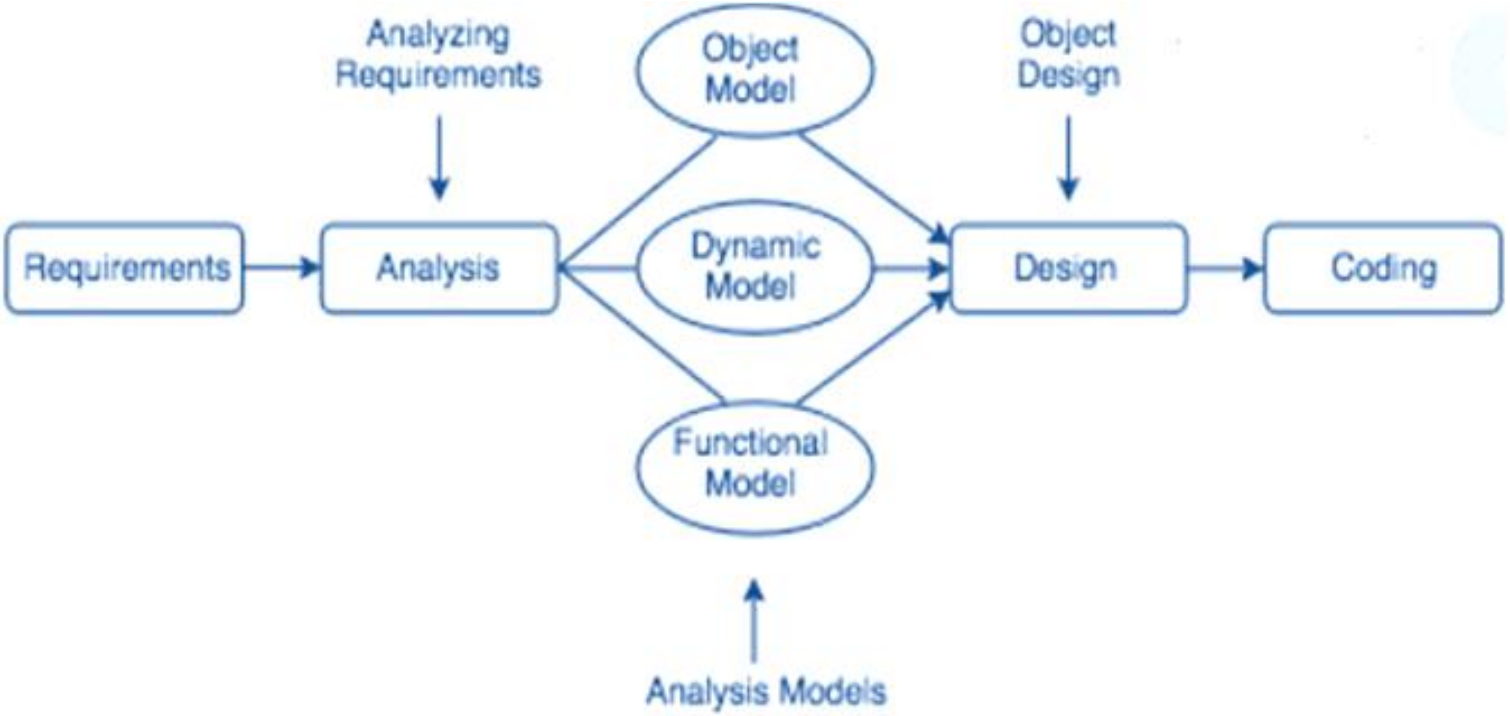
Rumbaugh Methodology

- Rumbaugh developed a technique that focuses on analysis, design and implementation of the system.
- This technique is popularly known as Object Modeling Technique (OMT).
- The OMT consists of four phases: analysis, system design, object design and implementation

❑ Analysis phase:

- Analysis phase is composed of three sub models given below:
- Object model
- Dynamic model
- Functional model

Object Oriented Methodologies



Object Oriented Methodologies

❑ **System design phase:**

In this phase high level design is developed taking the implementation environment including DBMS and communication protocols into account.

❑ **Object design phase:**

The goal of this phase is to define the objects in details.

The algorithms and operations of the objects are defined in this phase.

New objects may be identified to represent the intermediate functionality.

❑ **Implementation phase:**

Finally the objects are implemented following coding standards and guidelines.

Object Oriented Methodologies

Jacobson Methodology

- All the methodologies described above still lack of a comprehensive architecture to develop a software project.
- The Jacobson's methodology known as "Object Oriented Software Engineering (OOSE)" consists of five models:

❑ The requirement model:

The aim of the model is to gather software requirements.

❑ The analysis model:

The goal of this model is to produce ideal, robust and modifiable structure of an object.

Object Oriented Methodologies

- **The design model:**

  It refines the objects keeping the implementation environment in mind.

- **The implementation model**:

  It implements the objects.

- **The test model:**

  The goal of the test model is to validate and verify the functionality of the system.

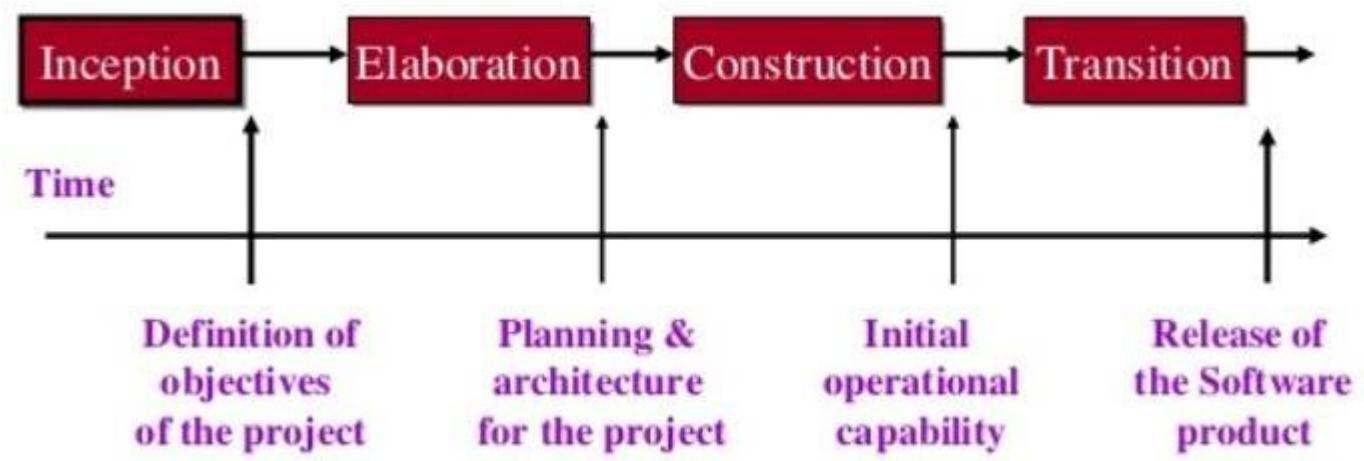
Object Oriented Methodologies

UML(Unified Modeling Language)

- UML represents the combination of the notations used by Booch, Rumbaugh and Jacobson.
- The best concepts and processes were extracted from all the methodologies till date and combined into UML.
- UML was adopted by Object Management Group (OMG) in November, 1997.
- UML is defined as language for visual modeling that allows to specify, visualize, construct, understand and document the various artifacts of the system.

Rational Unified Process(RUP)

- The main goal of RUP is to create high quality software with a predictable budget and time frame.



Vel Tech

Rangarajan Dr. Sagunthala

R&D Institute of Science and Technology

(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

Thank You

Department of Computer Science and Engineering