Shri Vaishnav Vidyapeeth Vishwavidyalaya

Department of Computer Science & Engineering

Subject:- SEPM

Subject Code:- BTCS504

Class:- 3Year CS - C, D, E & H

Ms. Mandakini Ingle
Assistant Professor
CSE

➢ UNIT–I

Nature of Software: Software Engineering, Software Process, A Generic Process Model, Process Assessment and Improvement, Prescriptive Process Models- Waterfall Model, Incremental Models, Evolutionary Models, Concurrent Models, Specialized Process Model, Unified Process, Personal and Team process Models, Process technology, Agile development.

What is Software?

- Collection of programs & related documents .
- The product that software professionals build and then support over the long term.

Software encompasses:

(1) instructions (computer programs) that when executed provide **desired features, function, and better performance**;

(2) data structures that enable the programs to adequately store and manipulate information and

(3) documentation that describes the operation and use of the programs.

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 Components

1. Programs

- Source Code & Object code
- Instant of software
- One part of entire software/Subset of software

2. Documents

- It consists various manuals [User manual & Operational manual]
- Cated at the end of each phase.

3. Procedures

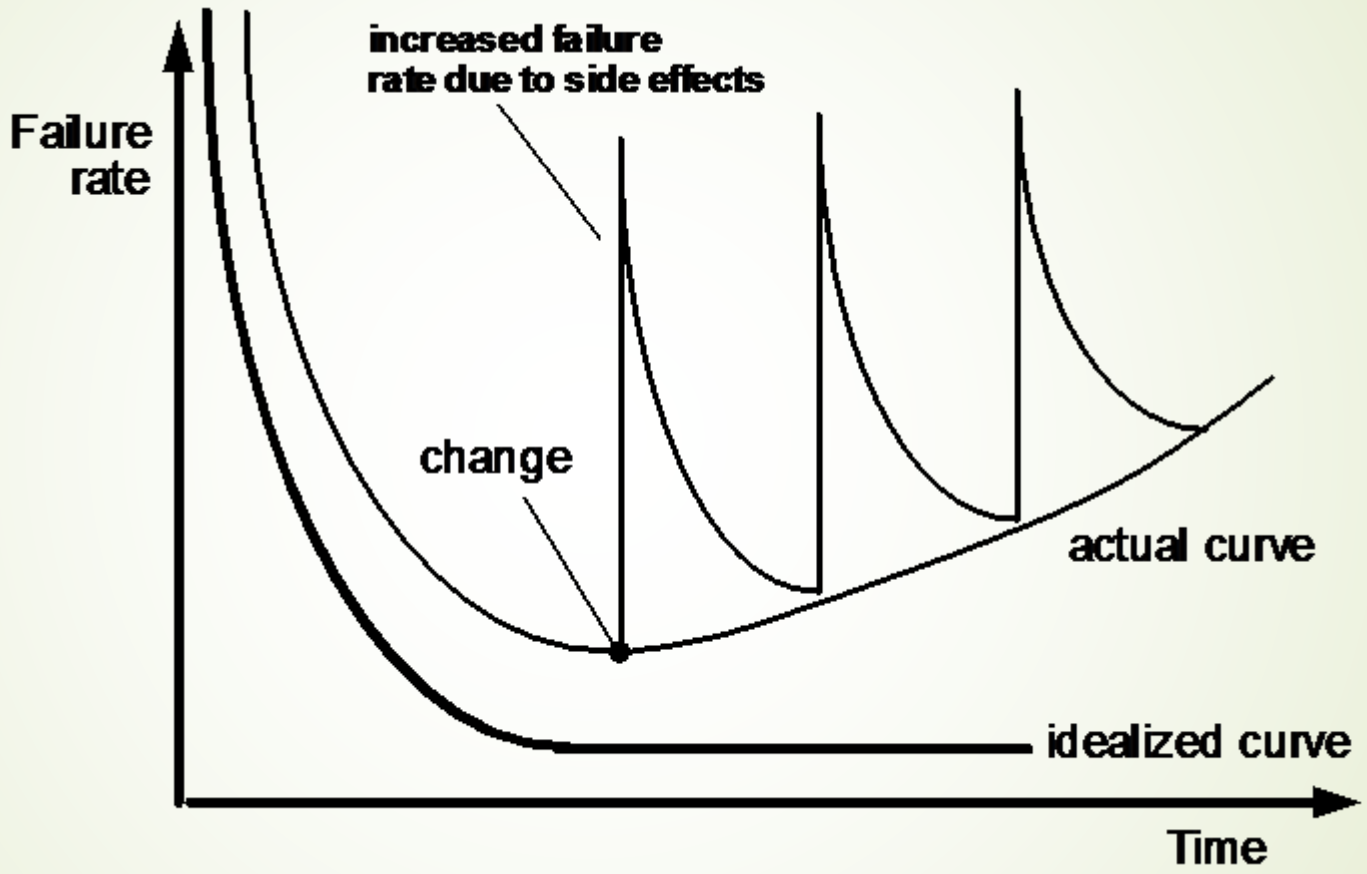
- Instruction for the initial setup & use of software system.
- Consists of some troubleshootinginstructions
- It can be given in manuals.

Characteristics / Features of Software

Features of such logical system:

- **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)**. Hardware has bathtub curve of failure rate (high failure rate in the beginning, then drop to steady state, then cumulative effects of dust, vibration, abuse occurs).

Wear vs. Deterioration



Characteristics / Features of Software.....

- Although the industry is moving toward component-based construction (e.g. standard screws and off-the-shelf integrated circuits), **most software continues to be custom-built**. Modern reusable components encapsulate data and processing into software parts to be reused by different programs. E.g. graphical user interface, window, pull-down menus in library etc.

Nature of Software

- ✓ Delivers Computing potential
- ✓ It resides on various resources
- ✓ Delivers imp product Time : Information
- ✓ Gateway to worldwide information
- ✓ Sophistication & complexity
- ✓ Adoption of software engg practice

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

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Software Engineering Definition

The seminal definition:

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The IEEE definition:

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Software Engineering Definition

- Software engineering is a descipline in which theories,methods & tools are applied to drevelop professional software product.

- Systematic & organized approach.

- Based on two terms

1. **Discipine**:- An engineer applies appropriate theories,methods & tools.Organizational & financial constraints.
2. **Products**:-

Software products

- Software when made for a specific reuirementsis called **software product.**
- Software products are a Software systems that delivered to customer with documentation.
- It may be a part of system product,whee hardware plus software delivered to customer.
- Software products are producedwith the help of the software process.
- It may be include source code , specification document,manuals documentation etc.

Software products Categories

1. Generic products

1. Stand-alone systems that are developed by production unit and sold to open market (**any customer)** who wishes to buy them.
2. Examples – PC software such as editing, graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

2. Customized products

1. Software that is commissioned by **a specific customer (software engineer)** to meet their own needs. Some constructor develops the software only for the customer.
2. Examples – embedded control systems, air traffic control software, traffic monitoring systems.

Characteristics of Software products

1. Efficiency
2. Maintinbility
3. Dependability
4. Ontime
5. Within Budget
6. Functionality
7. Adaptability

Software Process

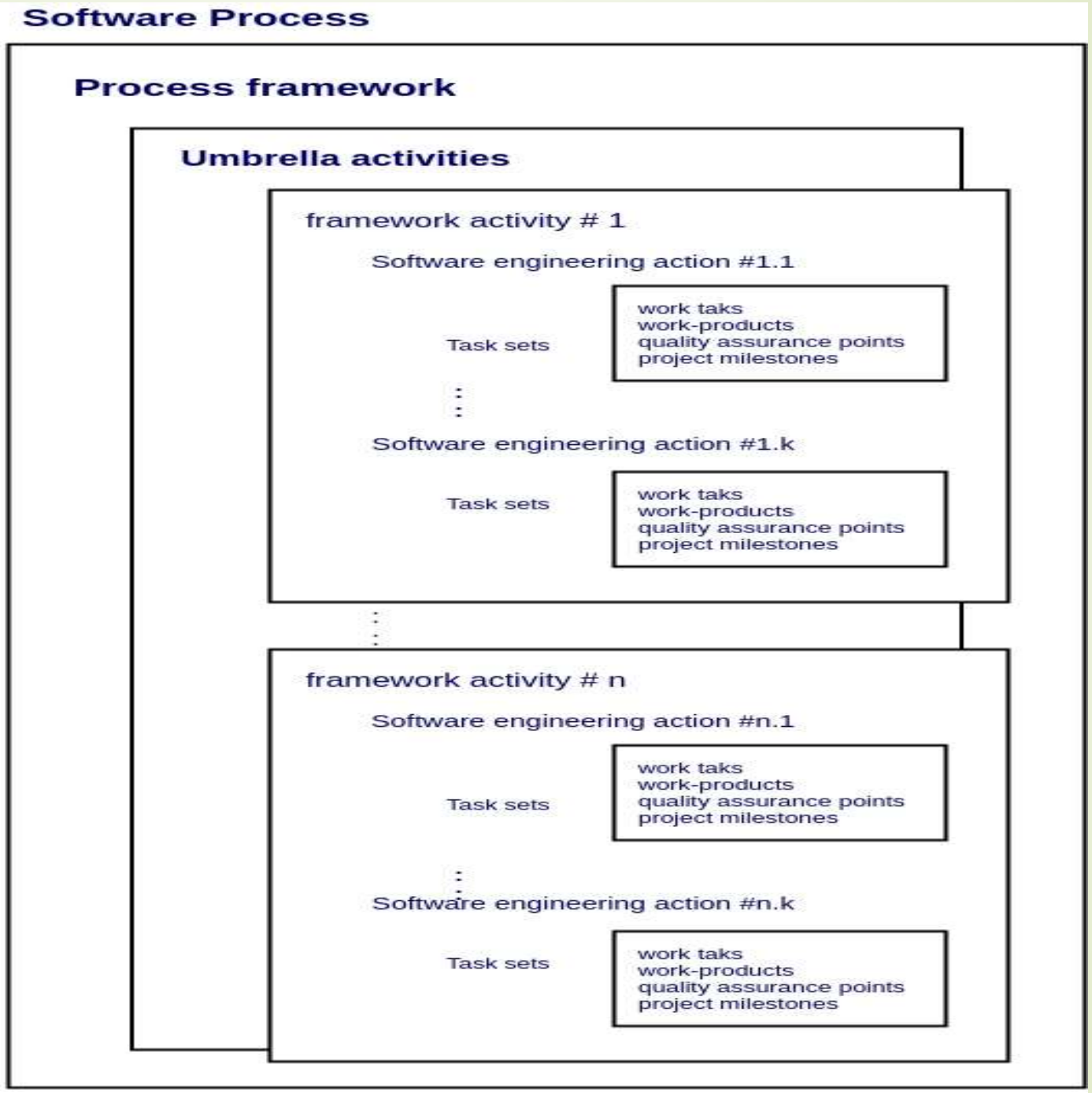
Software process can be defined as The structured **set of activities** that are required to develop the software system.

Fundamental activities are.....

- Specifications: The functionality of the software and constraints on its operation must be defined.
- Design & implementation: The software to meet the requirement must be produced.
- Validation: The software must be validated to ensure that it does what the customer wants.
- Evolution: The software must evolve to meet changing client needs.

- Software process model is an abstract representation of a process.
- It represent a description of a process from some particular perspective.

Common Software Process Framework



Common Software Process Framework

- Process framework is reuired for common process activity
- Software process is characterised by process framework activities, task sets & umbrella activity.

process framework activities.....

- **Communication:** communicate with customer to understand objectives and gather requirements
- **Planning:** Engineering plan,creates a "map" defines the work by describing the tasks, risks and resources, work products and work schedule.
- **Modeling:** Create a "sketch", what it looks like architecturally, how the constituent parts fit together and other characteristics.
- **Construction:** S/W design is mapped, code generation and the testing.
- **Deployment:** Delivered to the customer who evaluates the products and provides feedback based on the evaluation.

- These five framework activities can be used to all software development regardless of the application domain, size of the project, complexity of the efforts etc, though the details will be different in each case.
- For many software projects, these framework activities are applied **iteratively** as a project progresses. Each iteration produces a software increment that provides a subset of overall software features and functionality.

Common Software Process Framework....

1. FRAMEWORK ACTIVITIES:-

Task sets:- DefIne actual work done to achieve the s/w objectives.

- Collection of s/w engineering work tasks
- project milestones
- SQA

Umbrella Activities

Complement the five process framework activities and help team manage and control progress, quality, change, and risk.

1. **Software project tracking and control:** assess progress against the plan and take actions to maintain the schedule.
2. **Risk management:** assesses risks that may affect the outcome and quality.
3. **Software quality assurance**: defines and conduct activities to ensure quality.
4. **Technical reviews:** assesses work products to uncover and remove errors before going to the next activity.

Umbrella Activities.....

5. **Measurement:** define and collects process, project, and product measures to ensure stakeholder's needs are met.
6. **Software configuration management:** manage the effects of change throughout the software process.
7. **Reusability management:** defines criteria for work product reuse and establishes mechanism to achieve reusable components.
8. **Work product preparation and production:** create work products such as models, documents, logs, forms and lists.

Umbrella Activities.....

5. **Measurement:** define and collects process, project, and product measures to ensure stakeholder's needs are met.
6. **Software configuration management:** manage the effects of change throughout the software process.
7. **Reusability management:** defines criteria for work product reuse and establishes mechanism to achieve reusable components.
8. **Work product preparation and production:** create work products such as models, documents, logs, forms and lists.

A Layered Technology

- Any engineering approach must rest on organizational commitment to **quality** which fosters a continuous process improvement culture.
- **Process** layer as the foundation defines a framework with activities for effective delivery of software engineering technology. Establish the context where products (model, data, report, and forms) are produced, milestone are established, quality is ensured and change is managed.
- **Method** provides technical how-to's for building software. It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support.
- **Tools** provide automated or semi-automated support for the process and methods.

A Layered Technology

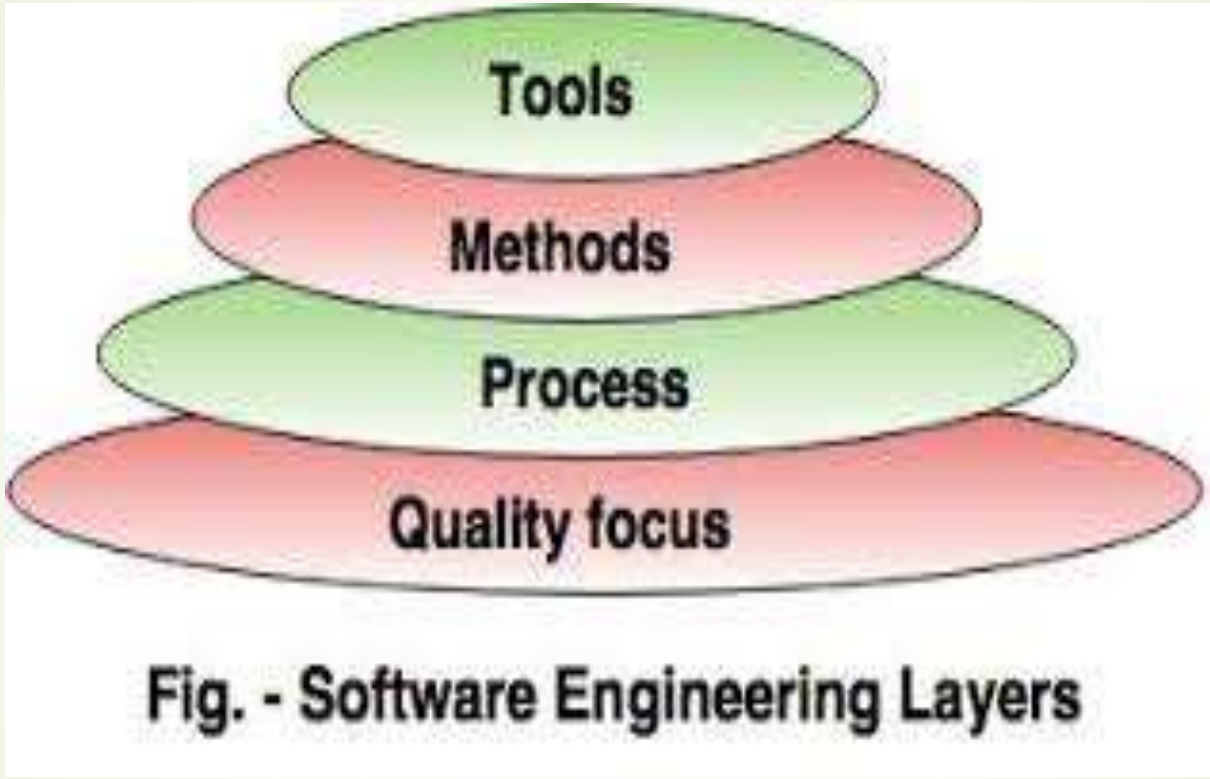


Fig. - Software Engineering Layers

A Layered Technology.....

A process is a collection of activities, actions and tasks that are performed when some work product is to be created. It is **not a rigid prescription** for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work to pick and choose the **appropriate set of work actions** and tasks.

Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

SOFTWARE PROCESS MODEL

- Abstraction of process
- Simplifiesd representation of software process.
- Each model represent a process from specific perspective.
- It represents the order in which the activities of s/w development will be undertaken
- Describe the seuence in which the phases ofs/w life cycle will be performed.
- It is a standarised format for Planning,organising & running to development of a project
- It is also called prodect life cycle/SDLC

SDLC

- SDLC stands for
  - Systems Development Life Cycle
  - SDLC is a *Life Cycle*.
  - All systems have a life cycle or a series of stages they naturally undergo.
    - The number and name of the stages varies, but the primary stages are conception, *development*, maturity and decline.
    - The systems development life cycle (SDLC) therefore, refers to the development stage of the system's life cycle.

SDLC

- The SDLC is a framework that describes the activities performed at each stage of a software development project.
- SDLC process is used by the software industry to design, develop and test high quality software.
- It aims to produce the quality software that meets or exceeds customer expectations, reaches completion within time and budget.

SDLC

- Every textbook has different names for the stages of the SDLC
  - Usually they stages are
    - Planning & Reuirement analysis (just after Conception)
    - Defining
    - Design
    - Building[Coding]
    - Testing
    - Depolyment
    - Maintenance (starting Maturity)


Maintenance
Requirement Analysis
Deployment
Defining
SDLC
(Software Development life Cycle)
Testing
Designing
Coding

SDLC

SDLC

- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.
- Software Engineering Process Technology Company, (SEPT) is a firm specializing in meeting the software process standards information needs of the professional community, particularly concerning ISO/IEC 12207.
- International Electrotechnical Commission (IEC)
- International Organization for Standardization (ISO)

SDLC

1. Planning & Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from all the stakeholders and domain experts or SMEs in the industry. Planning for the quality assurance requirements and identification of the risks associated with the project is also done at this stage

SDLC

Requirements Analysis • Business Requirements • Stakeholder Requirements • Solution Requirements Functional Requirements Non-functional Requirements • Transition Requirements

SDLC

2. Defining Requirements Once the requirement analysis is done the next step is to clearly define and document the software requirements and get them approved from the project stakeholders. This is done through 'SRS' – Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

SDLC

Defining Requirements • Enterprise Analysis •
Business Analysis Planning & Monitoring •
Elicitation • Requirements Analysis •
Requirements Management & Communication •
Solution Assessment & Validation

SDLC

3. Designing the Software

- Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.
- This DDS is reviewed by all the stakeholders and based on various parameters as risk assessment, design modularity , budget and time constraints , the best design approach is selected for the software.

SDLC

4. Developing the Software

• In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. • Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate and implement the code.

SDLC

5. Testing the Software

• This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. • However this stage refers to the testing only that stage of the software where defects are reported, tracked, fixed and retested, until the software reaches the quality standards defined in the SRS.

SDLC

6. **Deployment and Maintenance** • Once the software is tested and no bugs or errors are reported then it is deployed. • Then based on the feedback, the software may be released as it is or with suggested enhancements in the target segment. • After the software is deployed then its maintenance starts.

SDLC Models To help understand and implement the SDLC phases various SDLC models have been created by software development experts, universities, and standards organizations.

SDLC

Reasons for Using SDLC Models

- Provides the base for project planning, estimating & scheduling.
- Provides framework for standard set of terminologies, activities & deliverables.
- Provides mechanism for project tracking & control.
- Increases visibility of project progress to all stakeholders.

SDLC

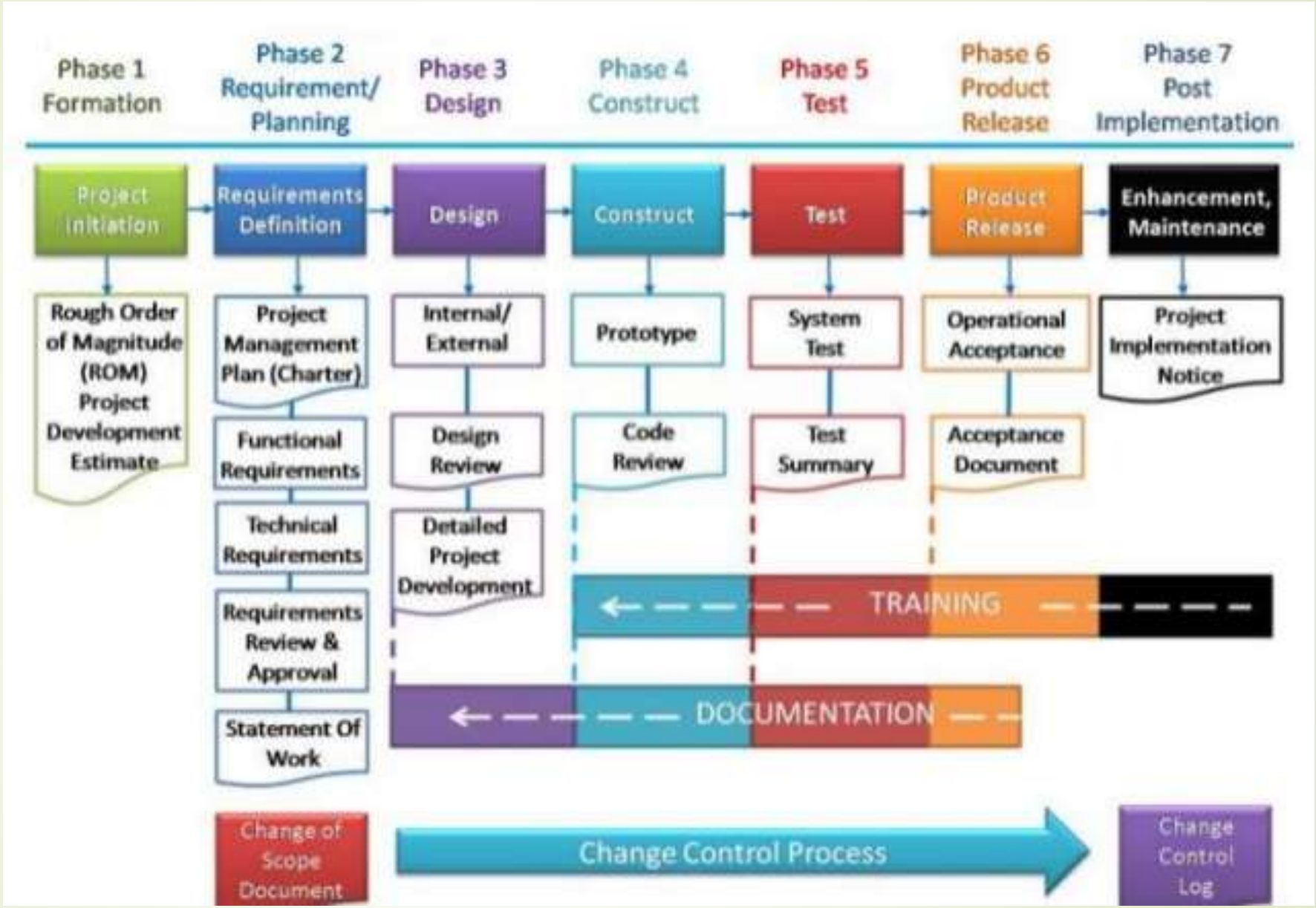
Advantages of Choosing an Appropriate SDLC

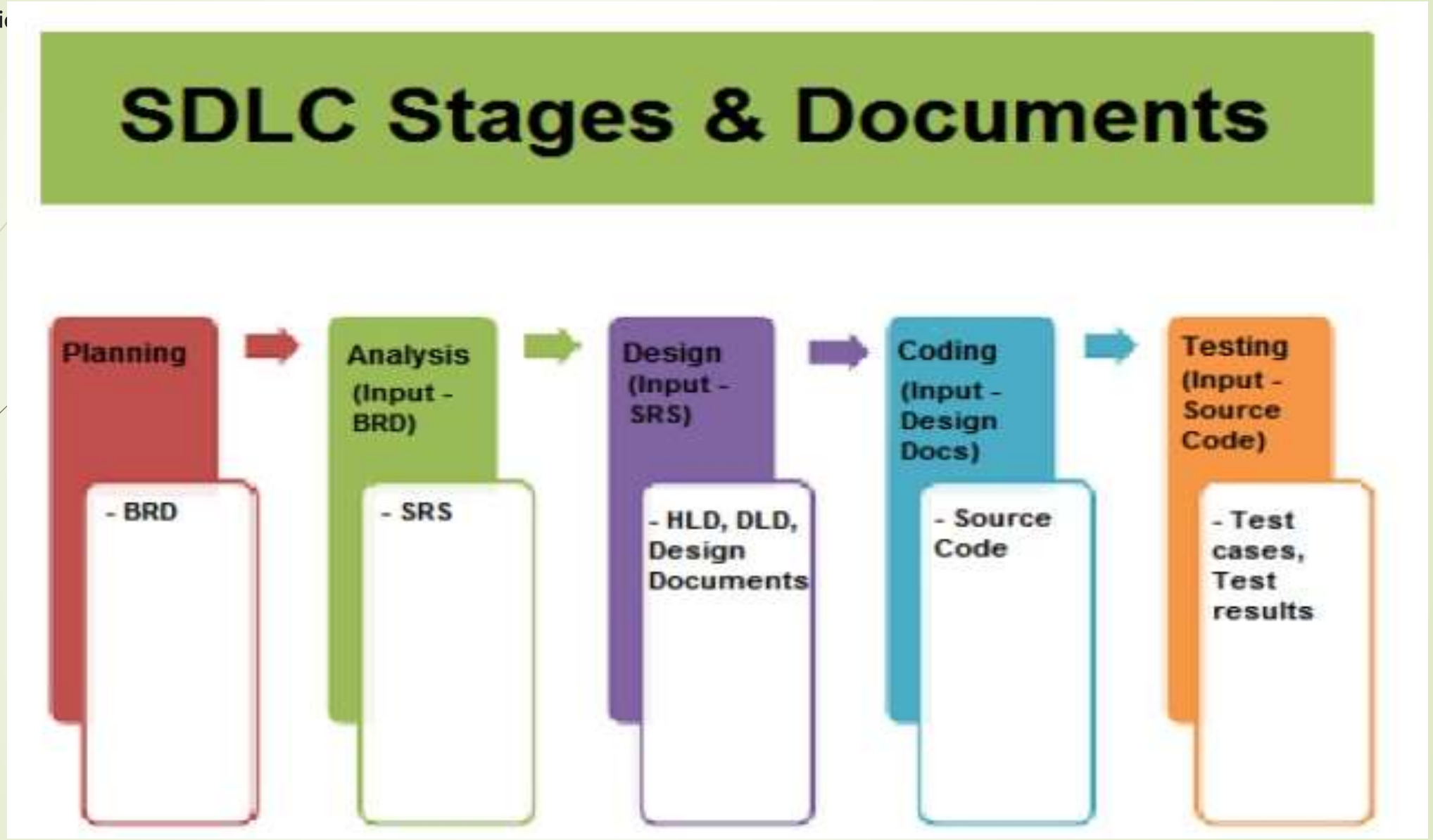
- Increased development speed
- Increased product quality
- Improved tracking & control
- Improved client relations
- Decreased project risk
- Decreased project management overhead

SDLC

SDLC Models

- Waterfall Model
- Iterative Model
- Spiral Model
- Agile Model
- V – Model
- Big Bang Model


Phase 1
Formation
Phase 2
Requirement/
Planning
Phase 3
Design
Phase 4
Construct
Phase 5
Test
Phase 6
Product
Release
Phase 7
Post
Implementation
Project Initiation
Requirements Definition
Design
Construct
Test
Product Release
Enhancement, Maintenance
Rough Order of Magnitude (ROM) Project Development Estimate
Project Management Plan (Charter)
Internal/ External
Prototype
System Test
Operational Acceptance
Project Implementation Notice
Functional Requirements
Design Review
Code Review
Test Summary
Acceptance Document
Technical Requirements
Detailed Project Development
Requirements Review & Approval
TRAINING
Statement Of Work
DOCUMENTATION
Change of Scope Document
Change Control Process
Change Control Log


SDLC Stages & Documents
Planning
- BRD
Analysis
(Input - BRD)
- SRS
Design
(Input - SRS)
- HLD, DLD, Design Documents
Coding
(Input - Design Docs)
- Source Code
Testing
(Input - Source Code)
- Test cases, Test results

Prescriptive Process Models

- Prescriptive process model were originally proposed to bring order to the chaos of software development.
- Prescriptive process model define a prescribed set of process elements and a predictable process work flow.
- “prescriptive” because they prescribe a set of process elements framework activities, software engineering actions, tasks, work products, quality assurance, and change control mechanisms for each project.

Prescriptive Process Models

The following framework activities are carried out irrespective of the process model chosen by the organization.

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

The name 'prescriptive' is given because the model prescribes a set of activities, actions, tasks, quality assurance and change the mechanism for every project.

Prescriptive Process Models

There are three types of prescriptive process models. They are:

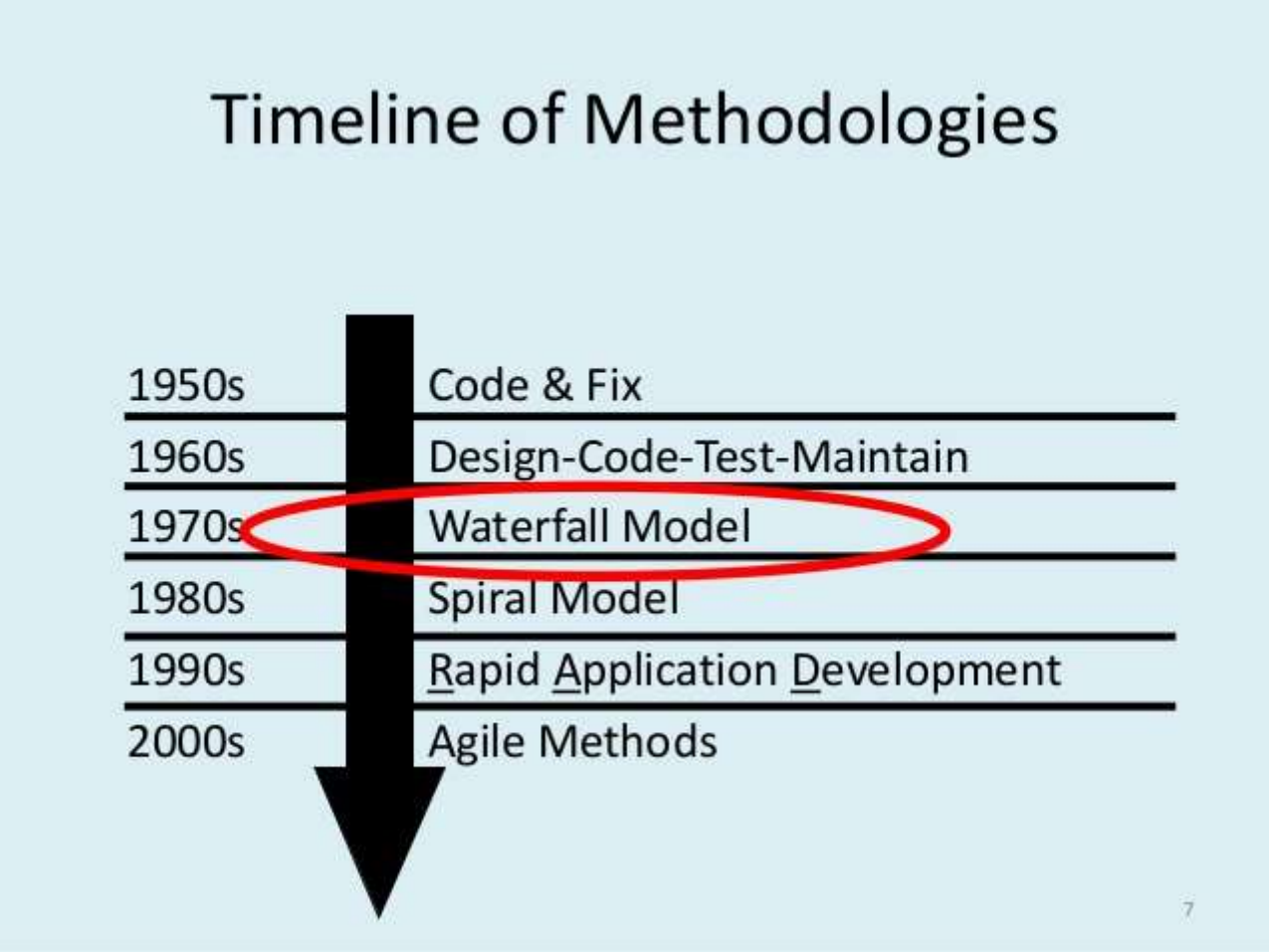
1. The Waterfall Model
2. Incremental Process model
3. RAD model

WATERFALL MODEL

Overview

- The "Waterfall Model" is a model that represents one method as to how software can be developed.

WATERFALL MODEL



WATERFALL MODEL

1)The first formal description of the waterfall model is often cited as a **1970 article by Winston W. Royce**

2)Royce did not use the term "waterfall" in this article.

3)Royce presented this model as an example of a flawed, non-working model.

WATERFALL MODEL

1) A Water Fall Model is easy to flow.

2) It can be implemented for any size of project.

3) Every stage has to be done separately at the right time so you cannot jump stages.

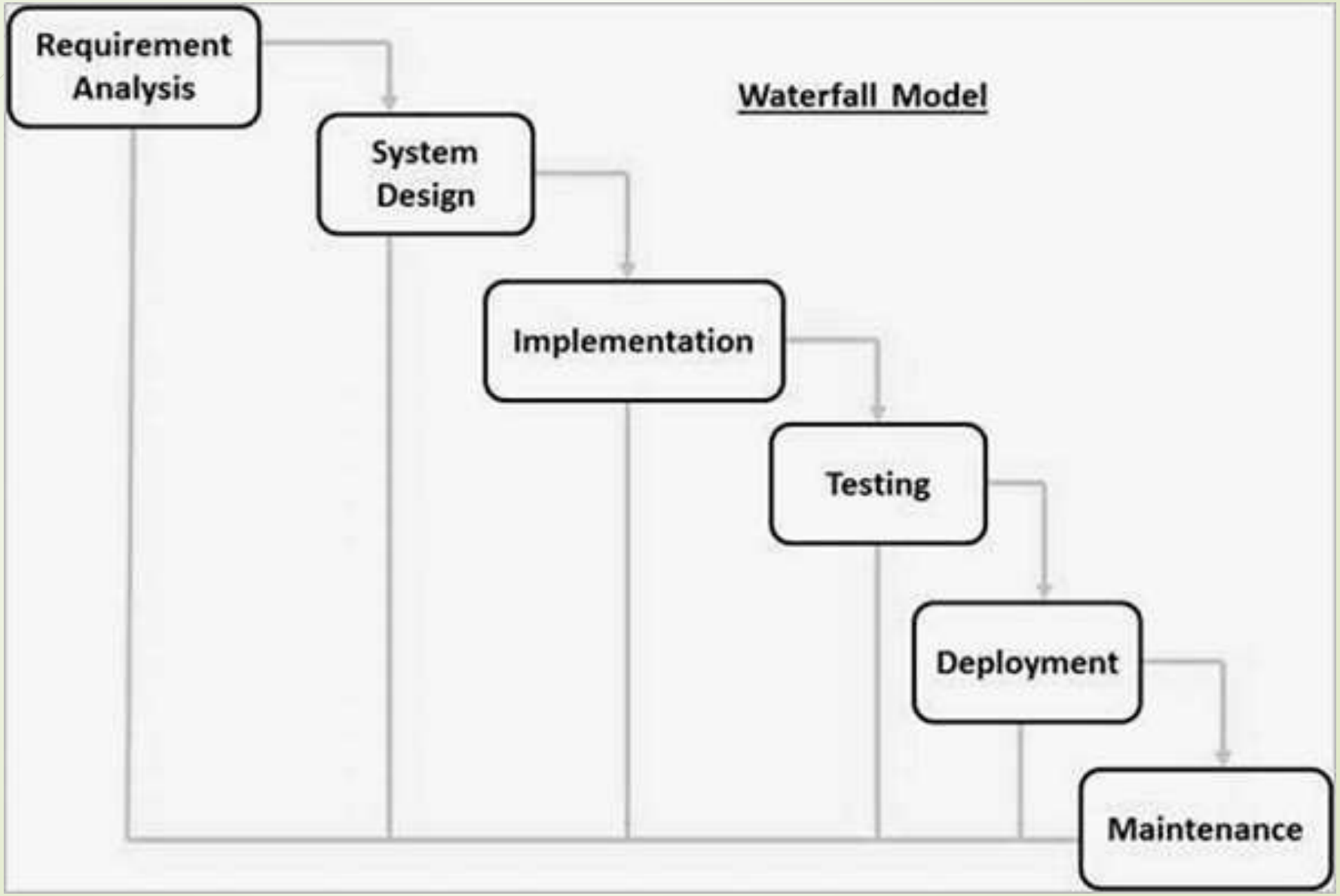
4) Documentation is produced at every stage of a waterfall model allowing people to understand what has been done.

5) Testing is done at every stage.

6) Linear Sequential Model/ Classical Life Cycle Model

It Is suggested systematic, sequential approach to software development.

WATERFALL MODEL



WATERFALL MODEL

1. Requirement gathering and Analysis.

- This is the first phase of waterfall model which includes a meeting with the customer to understand his requirements.
- This is the most crucial phase as any misinterpretation at this stage may give rise to validation issues later.
- The software definition must be detailed and accurate with no ambiguities.
- It is very important to understand the customer requirements and expectations so that the end product meets his specifications.

WATERFALL MODEL

1. Requirement gathering and Analysis. .....

- Requirement gathering and Analysis phase the basic requirements of the system must be understood by software engineer, who is also called ANALYST.

- All this requirements are then well documented and discussed further with the customer for reviewing.

WATERFALL MODEL

2)Design

- The customer requirements are broken down into logical modules for the ease of implementation.
- Hardware and software requirements for every module are Identified and designed accordingly.
- Also the inter relation between the various logical modules is established at this stage.
- Algorithms and diagrams defining the scope and objective of each logical model are developed.
- In short, this phase lays a fundamental for actual programming and implementation

WATERFALL MODEL

It is a intermediate step between requirements analysis and coding. Design focuses on program attribute such as-

1) Data Structure.

2) Software Architecture.

3) Algorithm Details

The requirements are translated in some easy to represent form using which coding can be done effectively and efficiently.

The desing needs to be documented for further use.

WATERFALL MODEL

3)Development / Coding

- Coding is a step in which design is translated into machine-readable form.
- If design is done in sufficient detail then coding can be done effectively. Programs are created in this phase.
- In this phase all software divided into small module then after doing coding for that small module rather than do coding whole software.
- According to design programmers do code and make class and structure of whole software.

WATERFALL MODEL

4)Testing

In this stage, both individual components and the integrated whole are methodically verified to ensure that they are error- free and fully meet the requirements outlined in the first step.

In this phase testing whole software into two parts

1) HARDWARE &

2) SOFTWARE.

Type of testing is 2-types 1) Inside test. 2) Outside test.

- Logical Internals of the Software.
- Uncover errors,fix the bugs & meet customers requirements

WATERFALL MODEL

5)Maintenance

- This is the final phase of the waterfall model, in which the completed software product is handed over to the client after alpha, beta testing.
- After the software has been deployed on the client site, it is the duty of the software development team to undertake routine maintenance activities by visiting the client site.
- If the customer suggests changes or enhancements the software process has to be followed all over again right from the first phase i.e requirement analysis.

WATERFALL MODEL

The usually the longest stage of the software.

In this phase the software is updated to:

a) Meet the changing customer needs

b) Adapted to accommodate changes in the external environment

c) Correct errors and oversights previously undetected in the testing phases

d) Enhancing the efficiency of the software Observe that feed back loops allow for corrections to be incorporated into the model.

WATERFALL MODEL ADVANTAGES

- The water fall model is easy to implementation.
- For implementation of small systems water fall model is use full.
- The project requires the fulfillment of one phase, before proceeding to the next.
- It is easier to develop various software through this method in short span of time.

WATERFALL MODEL DISADVANTAGES

- The requirement analysis is done initially and sometimes it is not possible to state all the requirement explicitly in the beginning.
- The customer can see working model of the project only at the end.
- If we want to go backtrack then it is not possible in this model.
- It is difficult to follow the sequential flow in software development process.
- Block States
-

RAD Model [*Rapid Application Development*]

1. Rapid application development is another form of incremental model.
2. It is a **high speed adaptation** of the linear sequential model in which fully functional system in a **very short time** (2-3 months).
3. This model is only applicable in the projects where requirements are well understood.
4. It employs more than one RAD teams and uses automatic code generation tools for modeling and construction of the application.

RAD Model.....

5. It focuses on input-output source and destination of the information.
6. It forces on delivering projects in small pieces; the larger projects are divided into a series of smaller projects.
7. The main features of RAD model are that it focuses on the reuse of templates, tools, processes and code.

RAD Model.....

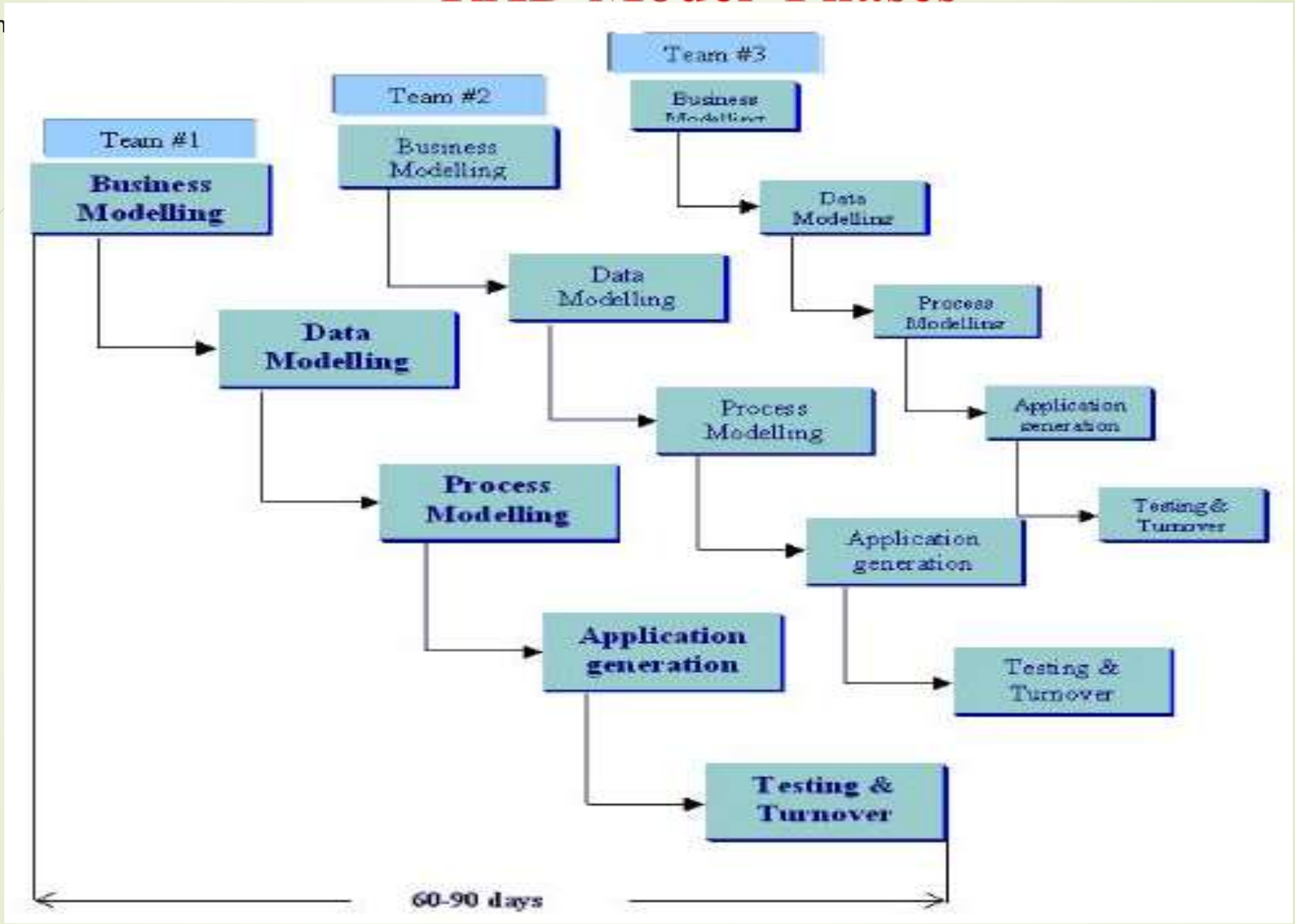
- In RAD model the components or functions are developed in parallel as if they were mini projects.
- The developments are time boxed, delivered and then assembled into a working prototype.
- A prototype is a working model that is functionally equivalent to a component of the product.
- This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

RAD Model

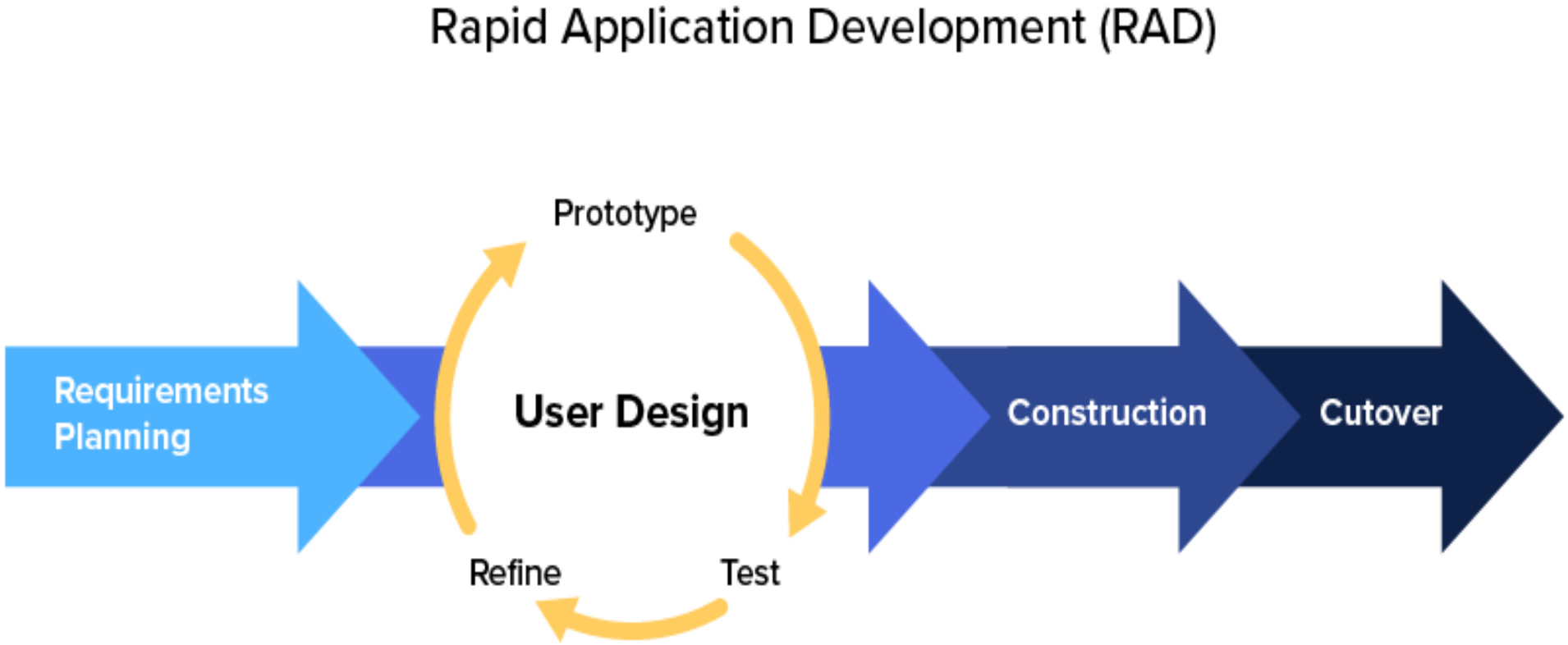
RAD model has following phases: □

1. Business Modeling □
2. Data Modeling □
3. Process Modeling □
4. Application Generation
5. Testing and Turnover

RAD Model Phases



RAD Model


Rapid Application Development (RAD)
Prototype
Requirements Planning
User Design
Construction
Cutover
Refine
Test

RAD Model

- **Business Modeling** On basis of flow of information and distribution between various business channels, the product is designed. Information flow is modelled into various business functions. business function collect information like...

1. That derive the business process
2. Type of information being generated
3. Generator of information
4. Information flow
5. Processors of information

RAD Model

- **Data Modeling** The information collected from business modeling is refined into a set of data objects that are significant for the business. Characteristics of data objects are identified. Relationship among various data objects is defined.

- **Process Modeling** The data object that is declared in the data modeling phase is transformed [PROCESS] to achieve the information flow necessary to implement a business function.

  Pocess are to extract the information from data objects & are resposible for implementing business function

RAD Model

- **Application Generation** Automated tools are used for the construction of the software, to convert process and data models into prototypes. Use of reusable components or create reusable componentsto have rpid development of S/W.

- **Testing & Turnover** As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.
- Testing efforts are reduceddue to reusable components.

RAD Model

Major Difference between RAD & Incremental Model

- RAD model is planned for short duration of implementation & Delivery.
- Most experienced programmers are involved in RAD Model.
- Incremental model Application will be developed in a set of divided timelines called iterations, each iteration results in deliverable product at the end.

RAD Model

- When to use RAD model When system needs to be produced in a short span of time (2-3 months).
- When the requirements are known.
- When the user will be involved all through the life cycle. When technical risk is less.
- When there is a necessity to create a system that can be modularized in 2-3 months of time.
- When budget is high enough to afford designers for modeling along with the cost of automated tools.

RAD Model Advantages

- Flexible and adaptable to changes.
- It is useful when you have to reduce the overall project risk. It is adaptable and flexible to changes.
- Due to code generators and code reuse, there is a reduction of manual coding.
- Due to prototyping in nature, there is a possibility of lesser defects.
- Each phase in RAD delivers highest priority functionality to client With less people, productivity can be increased in short time.

RAD Model Disadvantages

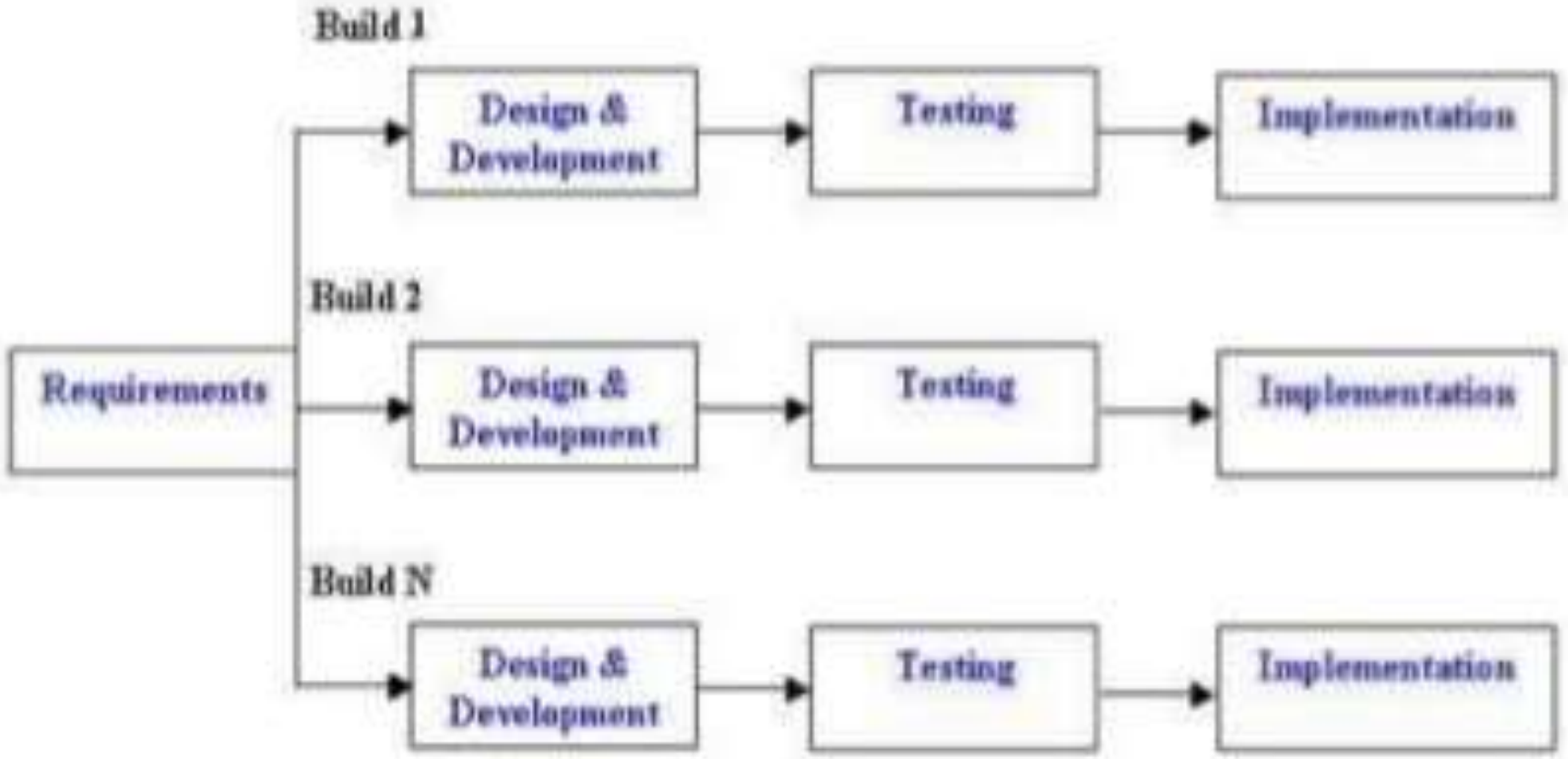
- It can't be used for smaller projects.
- Not all application is compatible with RAD.
- When technical risk is high, it is not suitable.
- If developers are not committed to delivering software on time, RAD projects can fail.
- Requires highly skilled designers or developers.
- Only system that can be modularized can be built using RAD.

Incremental model

- In incremental model the whole requirement is divided into various builds. LIke waterfall model.Analysis Design Code Test.
- Each module (independent units) passes through the requirements, design, implementation and testing phases.
- The incremental build model is a method of software development where the product is designed,
- implemented and tested incrementally until the product is finished.

- Each subsequent(coming after something in time) release of the module adds function to the previous release.
- The process continues till the complete system is achieved.

Incremental development model:



Incremental Life Cycle Model

Advantages of Incremental model:

- Generates working software quickly and early during the software life cycle.
- This model is more flexible, less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it's iteration.

Disadvantages of Incremental model:

- 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.

When to use the Incremental model:

- Mostly such model is used in web applications and product based companies.
- This model can be used when the 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.

EVOLUTIONARY MODEL

Evolutionary models are iterative. They are characterized in a manner that enables you to develop increasingly more complete versions of the software with each iteration. There are two common evolutionary process models.

Evolutionary Software Process Models

Classic process models are not designed to deliver a production system due to their assumptions on:

- *A complete system will be delivered after the linear sequence is completed.*
- *Customer knows what they want at the early stage.*

The realty in a software production process -->

- *A lot of requirements changes during the production course*
- *A lot of iterative activities and work because of the evolutionary nature of software production*

To cope with the product evolution, several evolution process models are proposed:

- *the incremental model*
- *the spiral model*
- *the component assembly model*

PROTOTYPE MODEL

Need of Prototype Model

- Even a small change in any previous stage can cause big problem for subsequent phases as all phases are dependent on each-other.
- Real projects rarely follow the sequential flow.
- Poor model for long and ongoing projects.
- WHY? As we freeze software and hardware in this model, but technology changes rapidly, such freezing is not advisable for long term projects.

PROTOTYPE MODEL

- Not a good model for complex and object-oriented projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- It is difficult to measure progress within stages.
- Based on the previous model disadvantages :-

PROTOTYPE MODEL

- This model requires before carrying out the development of the actual software, a working prototype of the system should be built.

- A prototype acts as a sample to test the process. From this sample we learn and try to build a better final product.

- A prototype is a model or a program which is not based on **strict planning**, but is an early **approximation of the final** product or software system.

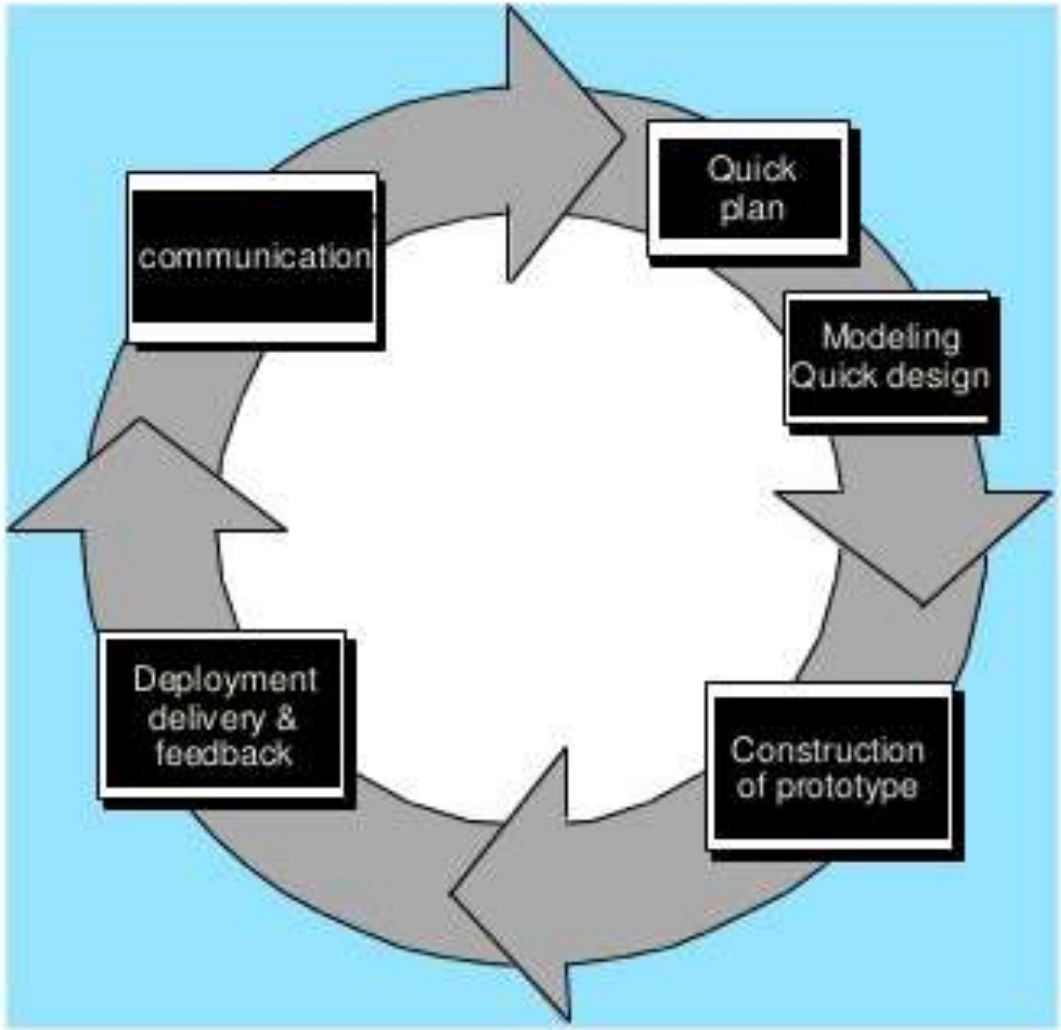
PROTOTYPE MODEL

- This type of System Development Method is employed when it is very difficult to obtain exact requirements from the customer (unlike waterfall model, where requirements are clear).

- While making the model, user keeps giving feedbacks from time to time and based on it, a prototype is made.

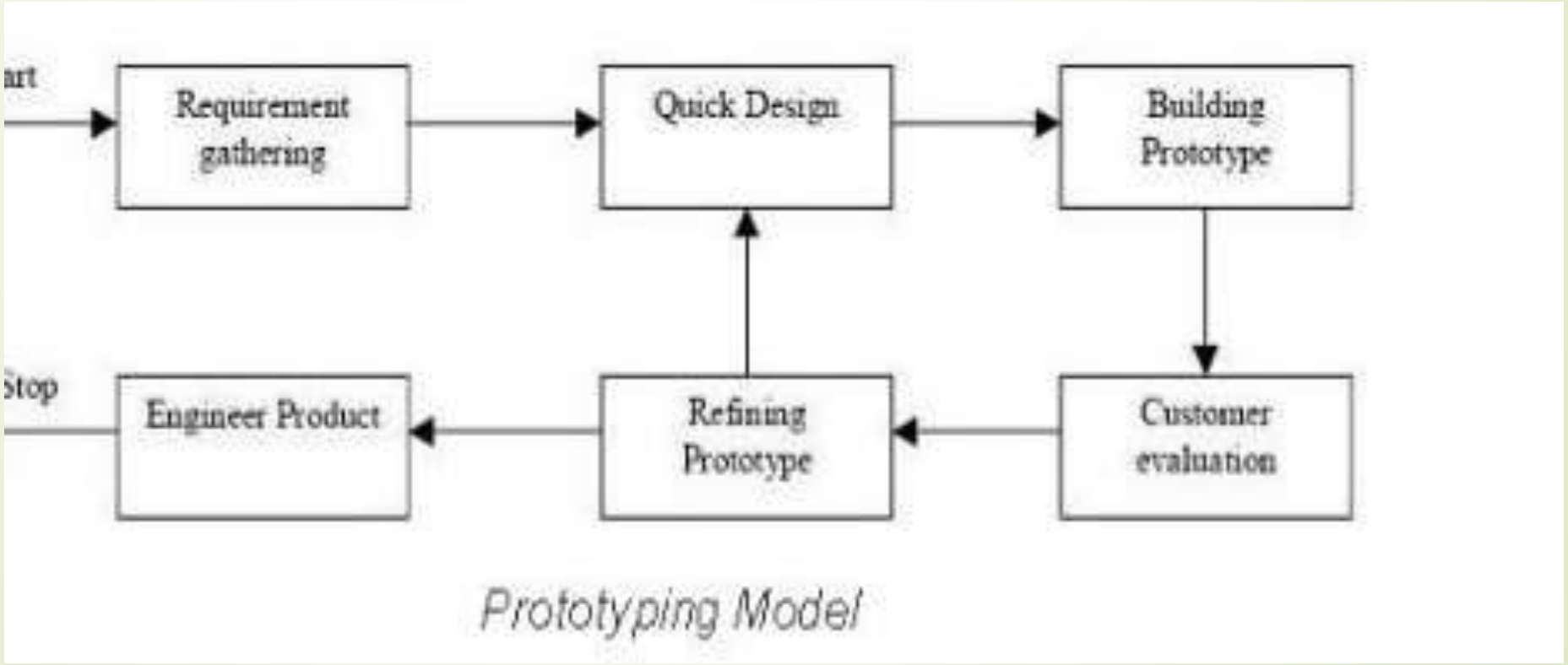
- Completely built sample model is shown to user and based on his feedback, the SRS (System Requirements Specifications) document is prepared.

Evolutionary Models: Prototyping



PROTOTYPE MODEL

- After completion of this, a more accurate SRS is prepared, and now development work can start using Waterfall Model, the goal is to provide a system with overall functionality.



Prototyping Model

PROTOTYPE MODEL ADVANTAGES

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Time required to complete the project after getting final the SRS reduces, since the developer has a better idea about how he should approach the project.

PROTOTYPE MODEL DISADVANTAGES

- It is a slow process.
- Too much involvement of client, is not always preferred by the developer.
- Too many changes can disturb the rhythm of the development team.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- If client is not satisfied with the developed prototype, new prototype should be made, this may lead to be expensive.

Spiral model:

- The spiral model is a system development lifecycle model.
- It allows refinement throughout each stage of the model.
- This means that one stage can be skipped for testing and returned to at a later point.
- Spiral model is an evolutionary software process model which is a combination of an iterative nature of prototyping and systematic aspects of traditional waterfall model.

Spiral model:

- The spiral model was defined by Barry Boehm in his 1988 article.
- This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters.

1. Concept developmen project
2. New product development projects
3. product inhancement projects
4. product maintenance projects

Each phase

Phases of Spiral model:

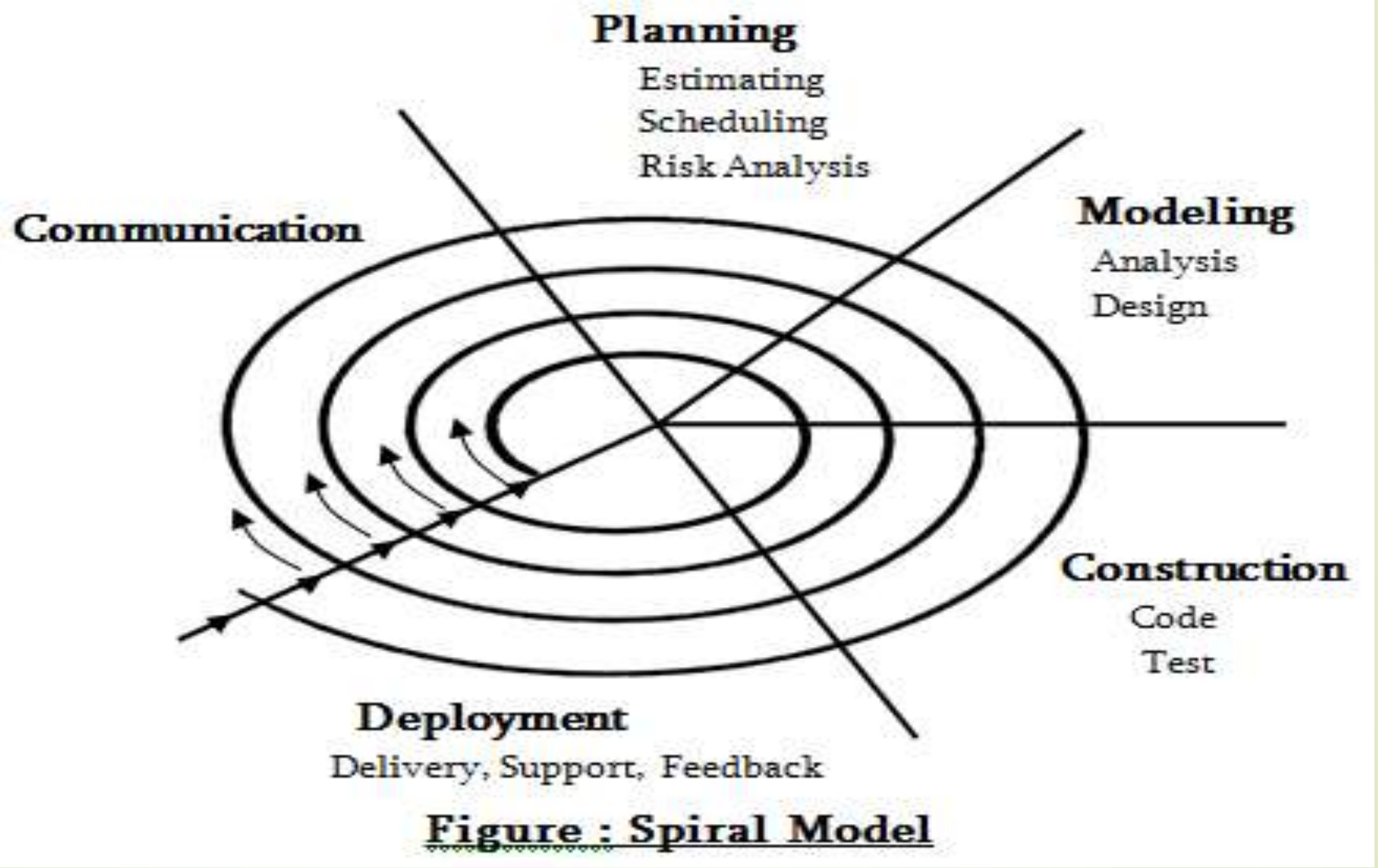


Figure : Spiral Model

Phases of Spiral Model (Tasks Regions)

1. **Customer communication** tasks required establishing effective communication between developer and customer.
2. **Planning**-tasks required defining resources, timelines, and other project related information.Estimation,Scheduling ,isk analysis.
3. **MOdeling analysis & Design** tasks required to assess both technical and management risks.
4. Engineering tasks required building one or more representations of the application.
5. **Construction and release** tasks required to construct, test, install, and provide user support.
6. **Customer evaluation-**tasks required to obtain customer feedback based on an evaluation of the software representations created during the engineering stage and implemented during the installation stage.

Advantages of the Spiral Model

A spiral model is a realistic approach to the development of large-scale systems and software. Check out some of its advantages:

1. 1. Estimates (i.e. budget, schedule, etc.) become more realistic as work progresses because more critical issues are discovered earlier.
2. It is more able to cope with the changes that software development entails.
3. Software engineers can get their hands in and start working on a project earlier.
4. Client can see the system early on
5. Changes in requirements can be made easily

Disadvantages of the Spiral Model

With advantages, there are also some disadvantages of the spiral model. Here are some of its disadvantages:

1. Highly customized limiting re-usability.
2. Applied differently for each application.
3. Risk of not meeting, budget or schedule.
4. Should not be used for small projects
5. This method requires a lot of documentation
6. Management is more complex

Applications of the Spiral Model

The spiral model is mostly used in large projects. The military had adopted the spiral model for its Future Combat Systems program.

The model thus may suit small software applications and not a complicated distributed, interoperable, system of systems.

When is best to use it?

Many reasons to use the spiral method are listed below:

- For high-risk projects
- The client is not sure about what requirements they want
- When the new product needs customer feedback to improve it
- Significant/major changes
- When costs and risk evaluation is important.
- For medium to high-risk projects.
- Users are unsure of their needs.
- Requirements are complex.
- Significant changes are expected.

Software Process customization improvement

- Software Process, also known as Software Development Process or Software Development Life Cycle.

- (SDLC) is simply dividing the software process into a set of activities or phases that lead to the production or development of software.

- These activities include designing, implementing, modifying, testing, maintaining, etc.

Software Process Customization .....

- These sets of activities are performed for improving the design, management of the project.
- Software process customization and improvement are also some techniques or activities that are performed to improve the overall development process.
- 1.Generic Software Product
- 2. Customized Software Product

Software Process Customization .....

Major Factor.....

1. People:-

- Product Successful or failure.

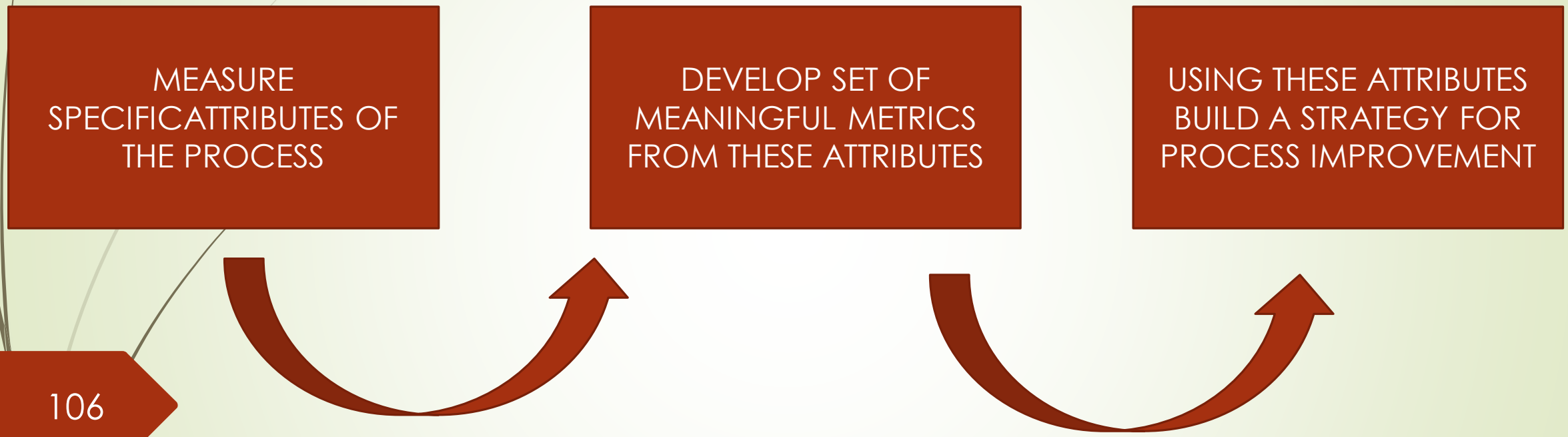
2. Product

- Scope & objective
- Technical or management contrains

3. Process

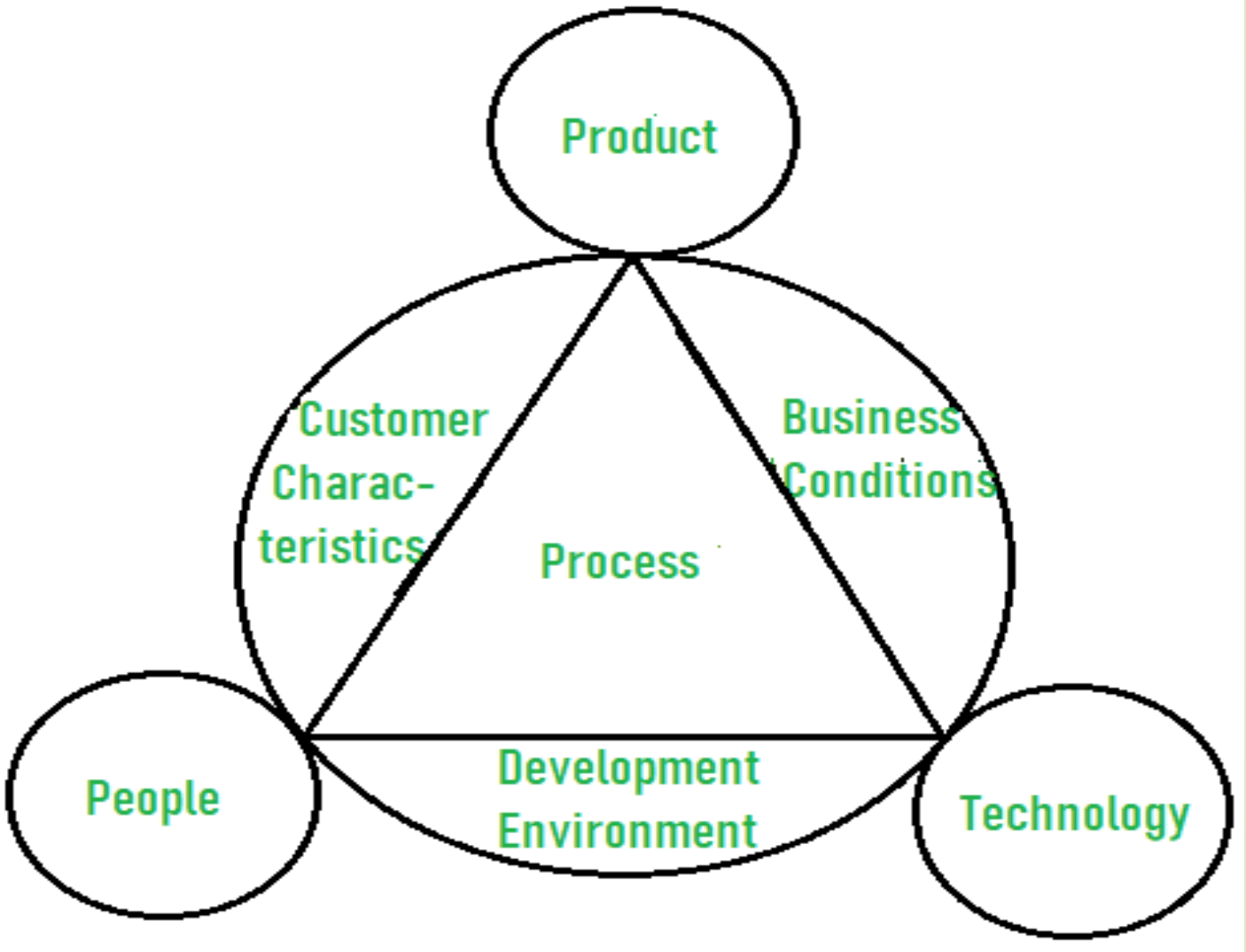
- s/w framework

Software Process Improvement.....



Software Process Improvement.....

- process metrics
- product
- people
- technology



Software Process Customization	Software Process Improvement
It is a process of designing, developing, deploying and even maintaining a software for a specific set of users.	It is simple defined as the definitions of sequence of various tasks, tools, and techniques that are needed to performed to plan and just implement all the improvement activities.
It includes three main factors: People, Product and Process .	It includes three factors: People, Technology and Product.
It also includes application customization, application modernization and application management.	It also includes improve planning, implementation, evaluation
It is very expensive, requires detailed planning and is time consuming.	It reduces the cost, increases development speed by installing tools that reduces the time and word done by humans or to automate the production process.

Software Process Customization	Software Process Improvement
It increases the level of productivity. It increases product quality.	It increases the level of productivity. It increases product quality.
It is created to fulfill user's requirements.	It is created to achieve the specific goals such as increasing the development speed, achieving the higher product quality and many more.
It is created by in-house development teams or third-party.	It mproves team performance by hiring best people

Process assessment and improvement

Software processes are assessed to ensure their ability to control the cost, time and quality of software. Assessment is done to improve the software process followed by an organization.

Software Process Improvement (SPI) Cycle includes:

- Process measurement
- Process analysis
- Process change

CMM

- CMM: Capability Maturity Model
- Developed by the Software Engineering Institute of the Carnegie Mellon University
- Framework that describes the key elements of an effective software process.

CMM

- Describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one.

- Provides guidance on how to gain control of processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence.

CMM Process Maturity Concepts

- **Software Process :-** set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, user manuals)

- **Software Process Capability :-** describes the range of expected results that can be achieved by following a software process
- means of predicting the most likely outcomes to be expected from the next software project the organization undertakes

CMM Process Maturity Concepts

- **Software Process Performance :-** actual results achieved by following a software process

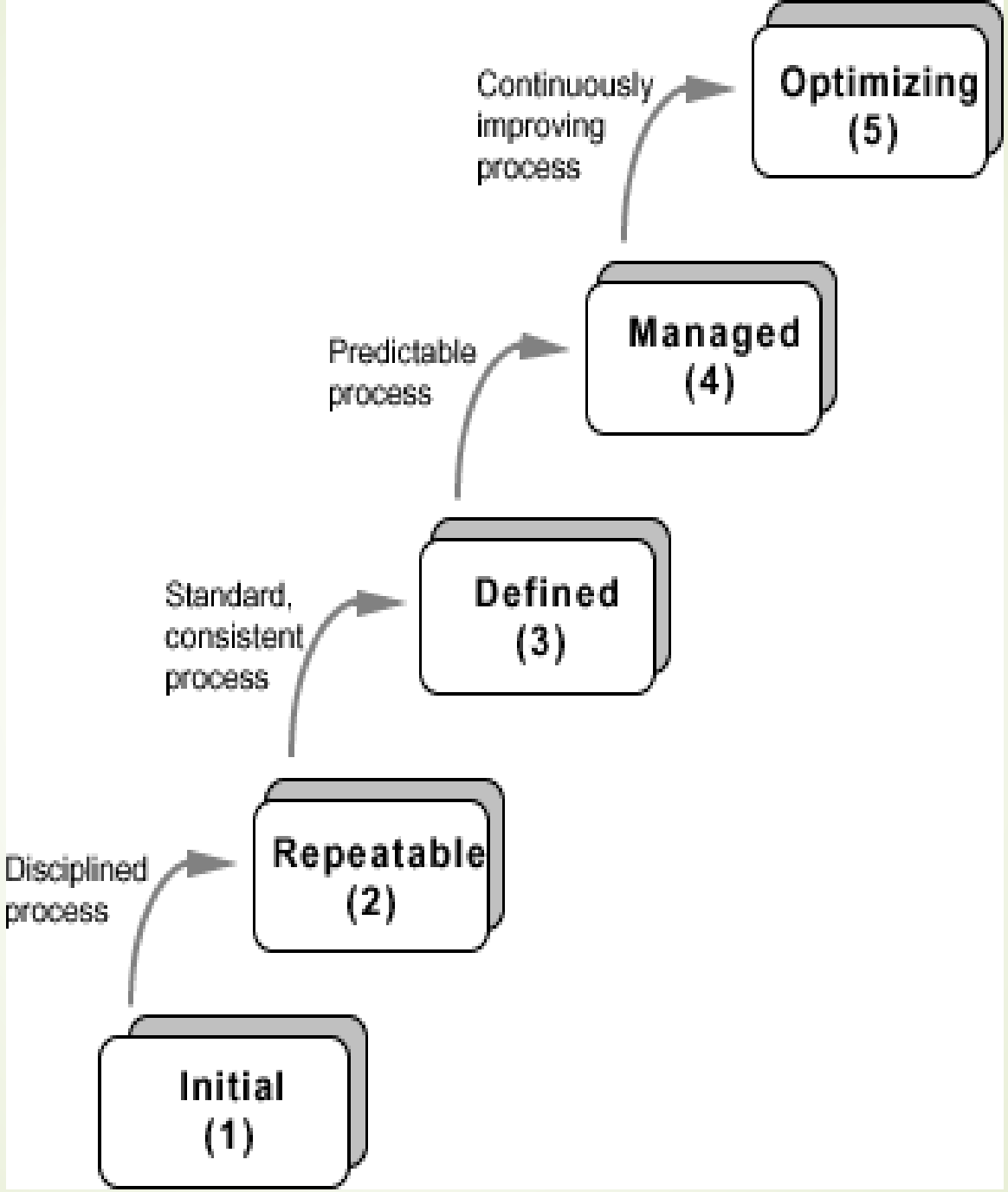
- **Software Process Maturity :-** extent to which a specific process is explicitly defined, managed, measured, controlled and effective
- implies potential growth in capability
- indicates richness of process and consistency with which it is applied in projects throughout the organization

The CMM Levels

The five levels of software process maturity

Maturity level indicates level of process capability:

- Initial
- Repeatable
- Defined
- Managed
- Optimizing


Optimizing
(5)
Continuously
improving
process
Managed
(4)
Predictable
process
Defined
(3)
Standard,
consistent
process
Repeatable
(2)
Disciplined
process
Initial
(1)

Level 1: Initial

- Initial : The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
  - At this level, frequently have difficulty making commitments that the staff can meet with an orderly process
  - Products developed are often over budget and schedule
  - Wide variations in cost, schedule, functionality and quality targets
  - Capability is a characteristic of the individuals, not of the organization

Level 4: Managed

- Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
  - Narrowing the variation in process performance to fall within acceptable quantitative bounds
  - When known limits are exceeded, corrective action can be taken
  - Quantifiable and predictable
    - predict trends in process and product quality

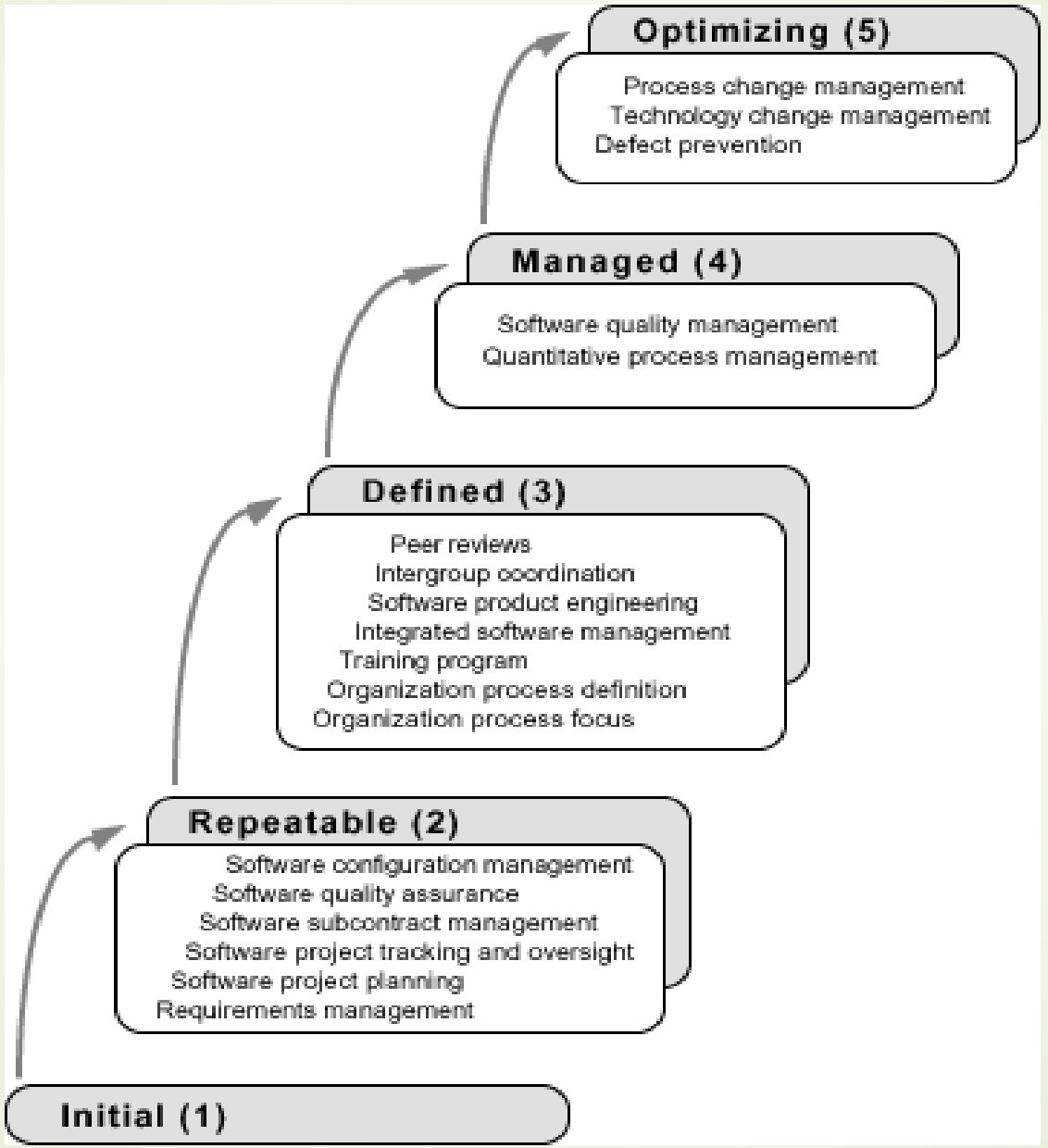
Level 5: Optimizing

- Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.
- Goal is to prevent the occurrence of defects
  - Causal analysis
- Data on process effectiveness used for cost benefit analysis of new technologies and proposed process changes

Internal Structure to Maturity Levels

- Except for level 1, each level is decomposed into key process areas (KPA)
- Each KPA identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing software capability.
  - commitment
  - ability
  - activity
  - measurement
  - verification

The Key Process Areas by Maturity Level



Level 2 KPAs

- **Requirements Management**
  - Establish common understanding of customer requirements between the customer and the software project
  - Requirements is basis for planning and managing the software project
  - Not working backwards from a given release date!
- **Software Project Planning**
  - Establish reasonable plans for performing the software engineering activities and for managing the software project

Level 2 KPAs

- Software Project Tracking and Oversight
  - Establish adequate visibility into actual progress
  - Take effective actions when project's performance deviates significantly from planned
- Software Subcontract Management
  - Manage projects outsourced to subcontractors
- Software Quality Assurance
  - Provide management with appropriate visibility into
    - process being used by the software projects

Level 2 KPAs

- Software Configuration Management
  - Establish and maintain the integrity of work products
  - Product baseline
  - Baseline authority

Level 3 KPAs

- Organization Process Focus
  - Establish organizational responsibility for software process activities that improve the organization's overall software process capability
- Organization Process Definition
  - Develop and maintain a usable set of software process assets
    - stable foundation that can be institutionalized
    - basis for defining meaningful data for quantitative process management

Level 3 KPAs

- **Training Program**
  - Develop skills and knowledge so that individual can perform their roles effectively and efficiently
  - Organizational responsibility
  - Needs identified by project
- **Integrated Software Management**
  - Integrated engineering and management activities
  - Engineering and management processes are tailored from the organizational standard processes
  - Tailoring based on business environment and project needs

Level 3 KPAs

- **Software Product Engineering**
  - technical activities of the project are well defined (SDLC)
  - correct, consistent work products
- **Intergroup Coordination**
  - Software engineering groups participate actively with other groups
- **Peer Reviews**
  - early defect detection and removal
  - better understanding of the products
  - implemented with inspections, walkthroughs, etc

Level 4 KPAs

- Quantitative Process Management
  - control process performance quantitatively
  - actual results from following a software process
  - focus on identifying and correcting special causes of variation with respect to a baseline process
- Software Quality Management
  - quantitative understanding of software quality
    - products
    - process

Level 5 KPAs

- Process Change Management
  - continuous process improvement to improve quality, increase productivity, decrease cycle time
- Technology Change Management
  - identify and transfer beneficial new technologies
    - tools
    - methods
    - processes
- Defect Prevention
  - causal analysis of defects to prevent recurrence

What are the benefits ?

- Helps forge a shared vision of what software process improvement means for the organization
- Defines set of priorities for addressing software problems
- Supports measurement of process by providing framework for performing reliable and consistent appraisals
- Provides framework for consistency of processes and product

Concurrent Models

The concurrent development model

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.
- The modeling activity completed its initial communication and then go to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modeling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state

Concurrent Models

- Concurrent models of software engineering involve multiple phases that can be executed at the same time. Learn the advantages, disadvantages, and applications of the waterfall, spiral, and prototype models used in software engineering.

Concurrent Models in Software

- Most of the successful software out there involves a series of phases of development, such as requirements gathering and prototyping, that are put together to develop the software. These phases are discrete and often performed concurrently. Often there is an intertwining between the phases, which makes it inevitable to return to the earlier phases to make some changes according to the results obtained in the later phases. This type of a model, in which multiple phases are performed concurrently, can be coined as a **concurrent model**. Some examples of concurrent models in software engineering will be discussed in this lesson.

Concurrent Models

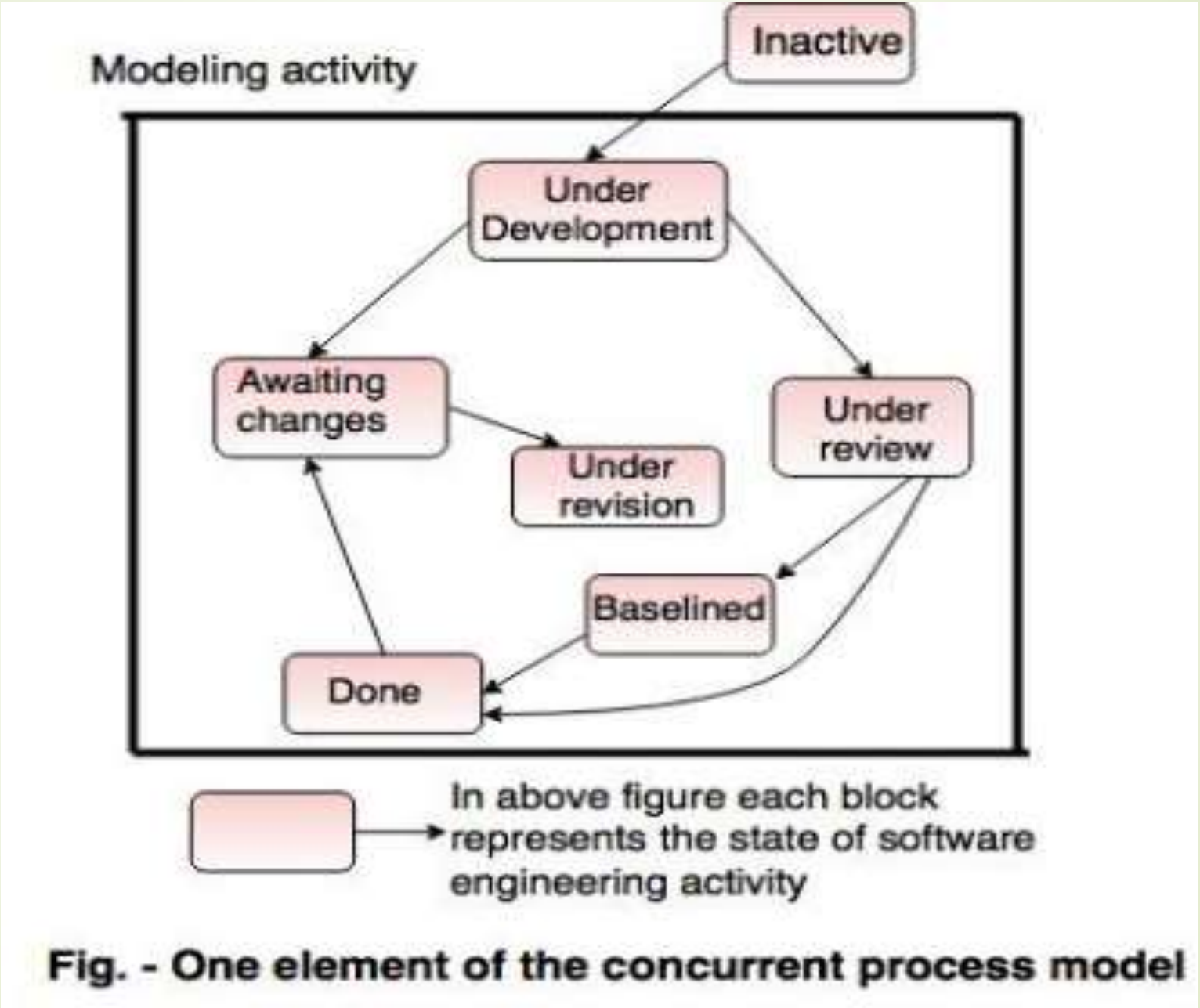


Fig. - One element of the concurrent process model

Concurrent Models

- For example,
- 'Design Phase' may be at an 'awaiting state' and 'customer communication' is in 'under revision' state. The customer wants some changes to the design, then 'communication' goes to 'awaiting changes' and 'design' goes to the under-development stage again.
- The benefit of this model is that project managers know each phase is what state and why.

Concurrent Models……

- **Advantages of the concurrent development model**
- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.
- **Disadvantages of the concurrent development model**
- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

Specialized Process Models.....

1. Component Based Development

- The component based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, Specialized process model demanding an iterative approach to the creation of software.
- However, the component based development model constructs applications from prepackaged software components.
- Modeling and construction activities begin with the identification of candidate components. These components can be designed as either conventional software modules or object oriented classes or packages of classes.
- Regardless of the technology that is used to create the components, the component based development specialized process model incorporates the following steps.

SPECIALIZED PROCESS MODELS

❑ Component-Based Development

- Provide targeted **functionality**
- Incorporates **characteristics** of spiral model
- Model **construct** applications
- **Researched** and evaluated for application
- Design to **accommodate** components
- **Integrated** into architecture
- **Testing** ensured proper functionality

Specialized Process Models.....

2. The Formal Methods Model

- The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software.
- Formal methods enable you to specify, develop, and verify a computer based system by applying a rigorous, mathematical notation.
- A variation on this approach, called clean room software engineering.
- When formal methods are used during development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms.
- **Ambiguity, incompleteness, and inconsistency** can be discovered and corrected more easily, but through the application of mathematical analysis.

Specialized Process Models……

2. The Formal Methods Model

- When formal methods are used during design, they serve as a basis for program verification and therefore enable you to discover and correct errors that might otherwise go undetected.
- Although not a mainstream approach, the formal methods model offers the promise of defect free software.
- **Draw Backs:**

- The development of formal models is currently quite time consuming and expensive.
- Because few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the models as a communication mechanism for Technically unsophisticated customers.

SPECIALIZED PROCESS MODELS

❑ The Formal Methods Model

- Leads to mathematical **specification**
- Mechanism for **eliminating** many problems
- Quite time **consuming** and **expensive**
- Difficult to use the models as **communication mechanism**

Specialized Process Models……

3. Aspect Oriented Software Development

- AOSD defines "aspects" that express customer concerns that cut across multiple system functions, features, and information.
- When concerns cut across multiple system functions, features, and information, they are often referred to as crosscutting concerns.
- Aspectual requirements define those crosscutting concerns that have an impact across the software architecture.
- Aspect oriented software development (AOSD), often referred to as aspect oriented programming (AOP), is a relatively new software engineering paradigm that provides a process and methodological approach for **defining, specifying, designing, and constructing aspects.**
- ” Grundy provides further discussion of aspects in the context of what he calls aspect oriented component engineering (AOCE):
- AOCE uses a concept of horizontal slices through vertically decomposed software components, called **“aspects,”** to characterize cross-cutting functional and non-functional properties of components.

SPECIALIZED PROCESS MODELS

❑ Aspect-Oriented Software Development

- Implement a set of localized feature, fun, info
- Modeled as a component (OO class)
- High level properties (eg. Security & fault tolerence)
- Sophisticated concern
- Crosscutting concern(fun, feature, info)

What is RUP?

- RUP was originally developed by Rational Software (now part of IBM).
  - It is a Software engineering process
  - It is a process product
  - It enhances team productivity
  - It creates and maintains models
  - It is a guide to effectively use the Unified Modeling Language
- Its goal is to delivery a high quality product that the customer actually wants.

- Stands for Rational Unified Process Framework
- Needs to be configured/tailored
- Contains phases, iterations, and workflows
- Helps an organization achieve CMM level 2/3 capabilities
- Improve existing processes based on proven Rational software engineering expertise
- Provide standards for requirements management
- Provide processes that support object-oriented analysis, design development for new J2EE efforts based on industry standard UML
- Well supported by industry leading Rational tools (Rose, RequisitePro)

Software like characteristics of RUP :

- Designed and documented using the Unified Modeling Language (UML).
- Delivered online using Web technology
- Software upgrades are released by Rational Software
- Modular and in electronic form, it can be tailored
- Integrated with the many software development tools

Why not use Waterfall instead?

- The Waterfall method follows a sequential approach to software development.
  - This limits the ability to react to any change or correct problems in a timely matter.
- Assumptions:
  - Requirements never change.
  - All information is known upfront.
  - The customer will be satisfied with the end results.
  - Technology will not change when it comes time to integrate.

One development cycle is divided into four consecutive phases.

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

-- **production phase**

- Each phase is concluded with a well-defined milestone
- Each phase has a specific purpose.

1. Inception –

1. Communication and planning are main.
2. Identifies Scope of the project using use-case model allowing managers to estimate costs and time required.
3. Customers requirements are identified and then it becomes easy to make a plan of the project.
4. Project plan, Project goal, risks, use-case model, Project description, are made.
5. Project is checked against the milestone criteria and if it couldn't pass these criteria then project can be either cancelled or redesigned.

Inception phase

- A vision document
- An initial use-case model
- An initial project glossary
- An initial business case
- An initial risk assessment
- A project plan, showing phases and iterations.
- A business model

- Project milestone: The Lifecycle Objectives Milestone

2. Elaboration –

1. Planning and modeling are main.
2. Detailed evaluation, development plan is carried out and diminish the risks.
3. Revise or redefine use-case model (approx. 80%), business case, risks.
4. Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be cancelled or redesigned.
5. Executable architecture baseline.

3. Construction –

1. Project is developed and completed.
2. System or source code is created and then testing is done.
3. Coding takes place.

Construction Phase

- The software product integrated on the adequate platforms.
- The user manuals.
- A description of the current release.

Project Milestone : Initial Operational Capability

4. Transition –

1. Final project is released to public.
2. Transit the project from development into production.
3. Update project documentation.
4. Beta testing is conducted.
5. Defects are removed from project based on feedback from public.

Transition Phase

- "Beta testing" to validate the new system against user expectations
- Parallel operation with a legacy system that it is replacing
- conversion of operational databases
- Training of users and maintainers
- Roll-out the product to the marketing, distribution, and sales teams

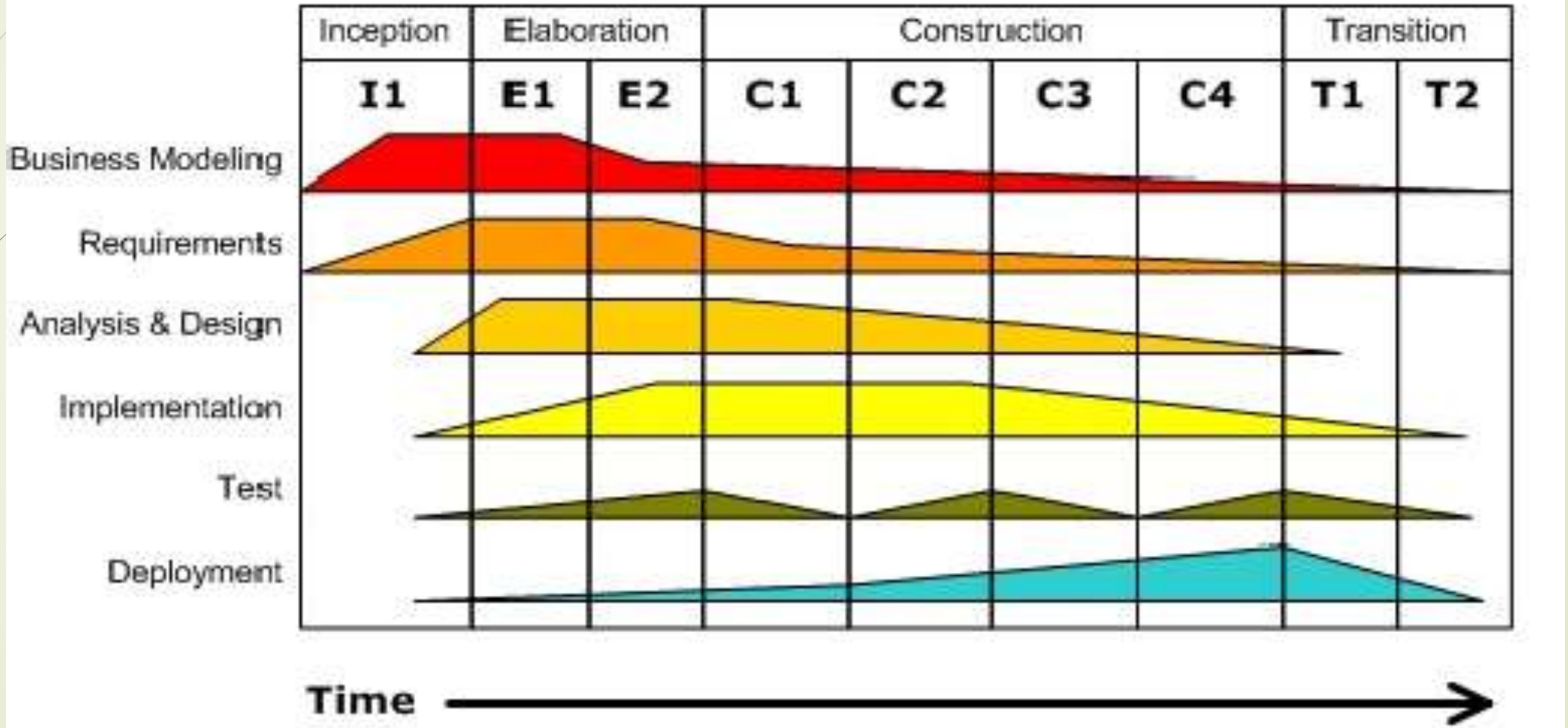
Project milestone: The Product Release Milestone

5. Production –

1. Final phase of the model.
2. Project is maintained and updated accordingly.

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



The Six Disciplines of RUP

1. Business Modeling
   - The goal is to understand the business of the organization.
2. Requirements
   - The goal is to define Scope: What is and is not to be built.
3. Analysis and Design
   - The goal is to analyze the requirements and design the solution.
4. Implementation
   - The goal is to execute the code based on the design.

The Six Disciplines of RUP (Cont.)

1. Test
   - The goal is to verify all aspects of the system to ensure quality.
2. Deployment
   - The goal is plan and deliver a working system to the customer.

Best Practices of the RUP

- Adapt the process
  - Adapt RUP appropriately based on the development needs.
- Balance competing stakeholders
  - Take an evolutionary approach by keeping stakeholders as active participants.
- Collaborating Across Teams
  - Keep an open communication process
- Demonstrate Value Iteratively
  - Deliver working software early and regularly

Best Practices of the RUP (Cont.)

- Elevate the level of Abstraction
  - Adapt modeling tools, reuse existing code, and focus on architecture
- Focus continuously on Quality
  - This is done by testing at every major part of the project.

Advantages of RUP

- Regular feedback from and to stakeholders
- Efficient use of resources
- You deliver exactly what the customer wants
- Issues are discovered early in your project
- Improved control
- Improved risk management

Disadvantages of RUP

- The process may be too complex to implement
- Development can get out of control
- It is a heavyweight process
- You need an expert to fully adopt this process

Iterative Approach

Compared to the traditional waterfall process, the iterative
the following advantages:

- Risks are mitigated earlier
- Change is more manageable
- Higher level of reuse
- The project team can learn along the way
- Better overall quality

Who Is Using the RUP?

- Telecommunications
- Transportation, aerospace, defense
- Manufacturing
- Financial services
- Systems integrators

Personal and Team Process Model

Personal and Team Process Model

- The best software process is personal and team process model one that is close to the people who will be doing the work. Watts Humphrey proposed two process models. Models "Personal Software Process (PSP)" and "Team Software Process (TSP)." Both require hard work, training, and coordination, but both are achievable.

Personal and Team Process Model

1, Personal Software Process (PSP)

The Personal Software Process (PSP) emphasizes personal measurement of both the work product that is produced and the resultant quality of the work product.

In addition PSP makes the practitioner responsible for project planning and empowers the practitioner to control the quality of all software work products that are developed. PSP stresses the need to identify errors early and, just as important, to understand the types of errors that you are likely to make. PSP represents a disciplined, metrics based approach to software engineering that may lead to culture shock for many practitioners. The PSP model defines five framework activities:

•***Planning.*** This activity isolates requirements and develops both size and resource estimates. In addition, defects estimate (the number of defects projected for the work) is made. All metrics are recorded on worksheets or templates. Finally, development tasks are identified and a project schedule is created.

Personal and Team Process Model

- ***High level design.*** External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.
- ***High level design review.*** Formal verification methods are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.
- ***Development.*** The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.
- ***Postmortem.*** Using the measures and metrics collected, the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness.

Personal and Team Process Model

Humphrey defines the following objectives for TSP:

- *Build self directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPTs) of 3 to about 20 engineers.*
- *Show managers how to coach and motivate their teams and how to help them sustain peak performance*
- *Accelerate software process improvement by making CMM23 Level 5 behavior normal and expected.*
- *Provide improvement guidance to high-maturity organizations.*
- *Facilitate university teaching of industrial-grade team skills.*

TSP defines the following framework activities: project launch, high level design, implementation, personal and team process model, integration and test, and postmortem.

Personal and Team Process Model

2. Team Software Process (TSP)

Watts Humphrey extended the lessons learned from the introduction of PSP and proposed a Team Software Process (TSP). The goal of TSP is to build a "self directed" project team that organizes itself to produce high quality software.

PERSONAL AND TEAM PROCESS MODELS

Personal Software Process (PSP)

The PSP model defines five framework activities

- ✓ Planning
- ✓ High-level design
- ✓ High-level design review
- ✓ Development
- ✓ Postmortem

Team Software Process (TSP)

- Build self directed teams that *plan and tract the work*
- Integrated product *team 3 to 20 enggs*
- Show managers how to *motivate their teams*
- Accelerate software *process improvements*
- Facilitate university *teaching of industrial team grade skills*
- Define *roles and responsibility* for each team members
- Track, manages and report *project status*

Agile Model

- Agile is a term used to describe approaches to software development emphasizing incremental delivery, team collaboration, continual planning, and continual learning, instead of trying to deliver it all at once near the end.

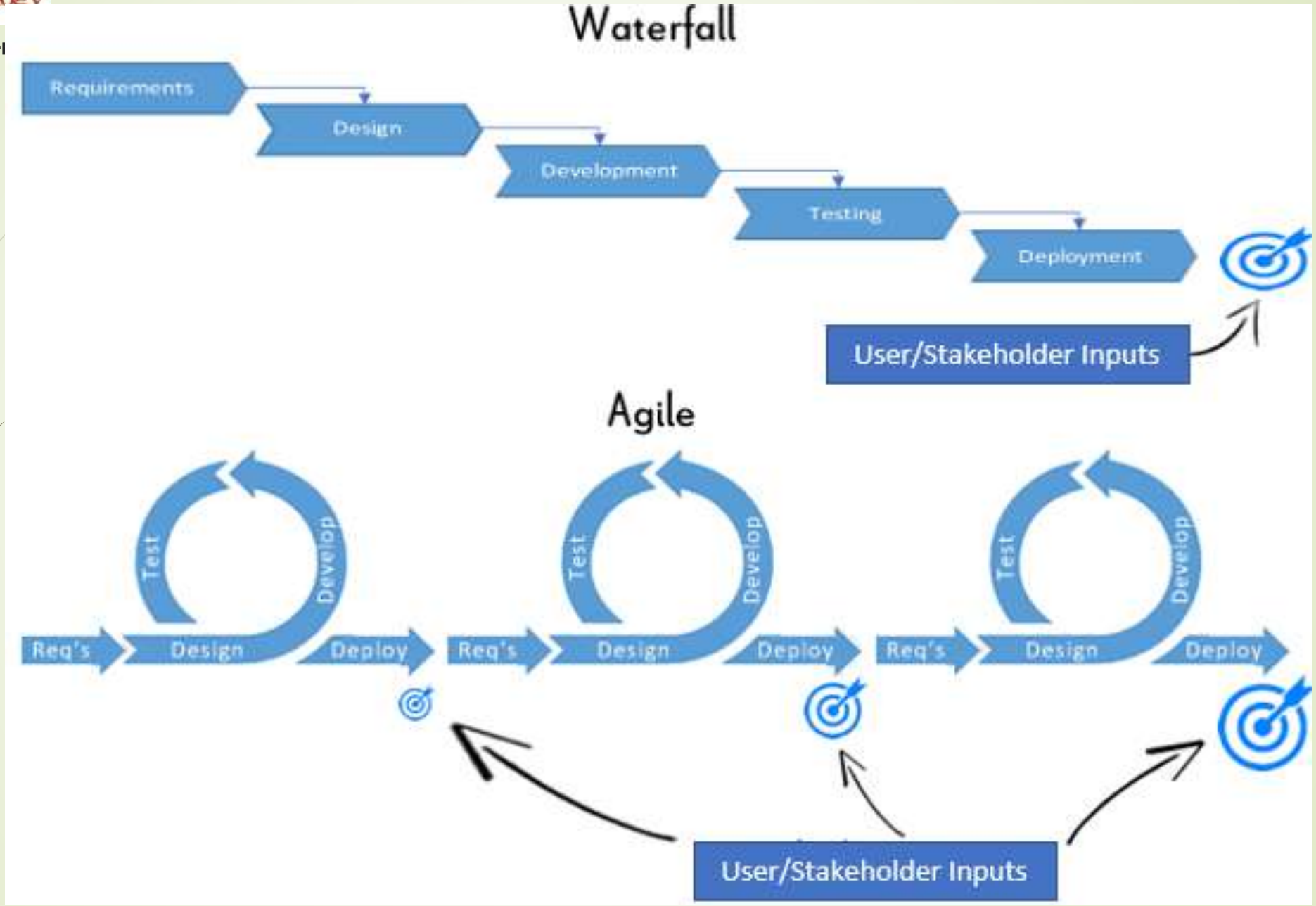
- Agile focuses on keeping the process lean and creating minimum viable products (MVPs) that go through a number of iterations before anything is final. Feedback is gathered and implemented continually and in all, it is a much more dynamic process where everyone is working together towards one goal.

Agile Model

- The meaning of Agile is swift or versatile."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 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.


Waterfall
Requirements
Design
Development
Testing
Deployment
User/Stakeholder Inputs
Agile
Test
Develop
Req's
Design
Deploy
Test
Develop
Req's
Design
Deploy
Test
Develop
Req's
Design
Deploy
User/Stakeholder Inputs

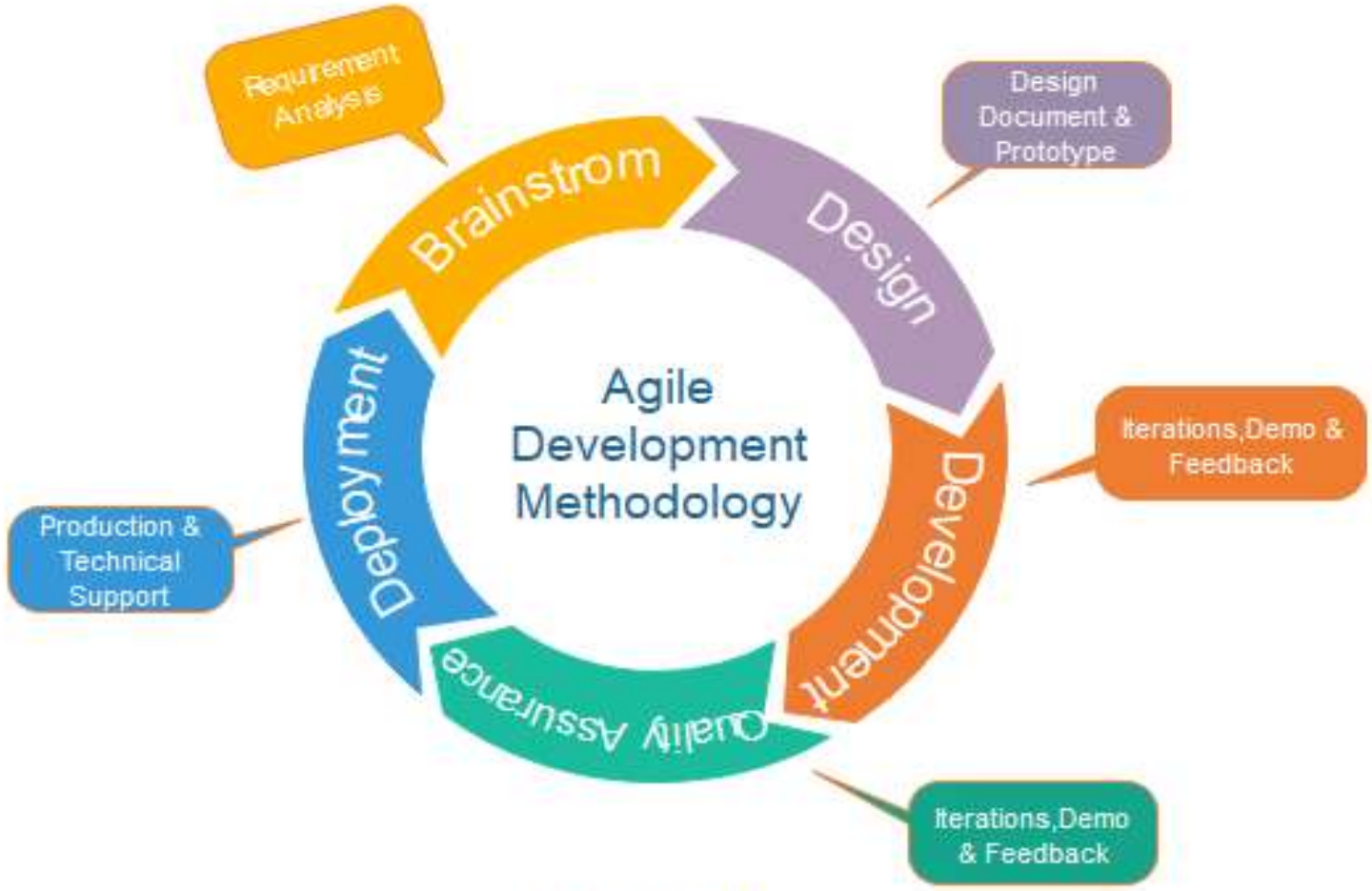


Fig. Agile Model

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

- Requirements gathering
- Design the requirements
- Construction/ iteration
- Testing/ Quality assurance
- Deployment
- Feedback

1. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Agile Testing Methods:

1. Scrum
2. Crystal
3. Dynamic Software Development Method(DSDM)
4. Feature Driven Development(FDD)
5. Lean Software Development
6. eXtreme Programming(XP)

When to use the Agile Model

1. When frequent changes are required.
2. When a highly qualified and experienced team is available.
3. When a customer is ready to have a meeting with a software team all the time.
4. When project size is small.

Disdvantage(Pros) of Agile Model

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

Advantage(Pros) of Agile Model

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

Agile Manifesto

The term "Agile" was coined in 2001 in the Agile Manifesto. The manifesto set out to establish principles to guide a better approach to software development.The Agile Manifesto consists of 4 important values. The way to read the Agile Manifesto is not that the items on the right side have no value anymore, but the Agile movement values the items on the left more.

Individual and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

AGILE PROCESS MODELS

- Agile process engineering combines a philosophy and a set of **development guidelines**.

- The philosophy encourages **customer satisfaction** and early incremental delivery of software, small highly motivated project teams, informal methods, minimal software engineering products, and overall development simplicity.

- The development guidelines stress delivery over analysis and design and active and continuous **communication between developer and customer**

Agility Principles

The Agile Alliance defines agility principles for those who want to achieve agility:

- Our highest priority is to satisfy the customer through **early and continuous delivery** of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's **competitive advantage**.
- **Deliver working software** frequently, from a couple of weeks to a couple of months, with a preference to the **shorter timescale.**
- Business people and developers must work **together daily throughout the project.**

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

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

- Working software is the primary measure of **progress.**

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

- **Continuous attention** to technical excellence and good design enhances agility.

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

The Politics of Agile Development

- This methodology debate risk degenerating into a religious war.
- The real que is: what is the best to achieve it
- Each model there is a set of ideas
- Many agile concepts are simply adaptations of good s/w engg.

ANY QUERY