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.

SOFTWARE PRODUCT AND PROCESS CHARACTERISTICS:

Software: -
Software is nothing but collection of computer programs and related documents that are planned to provide desired features, functionalities, and better performance.
Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

- **Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.
- 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.

Nature of Software

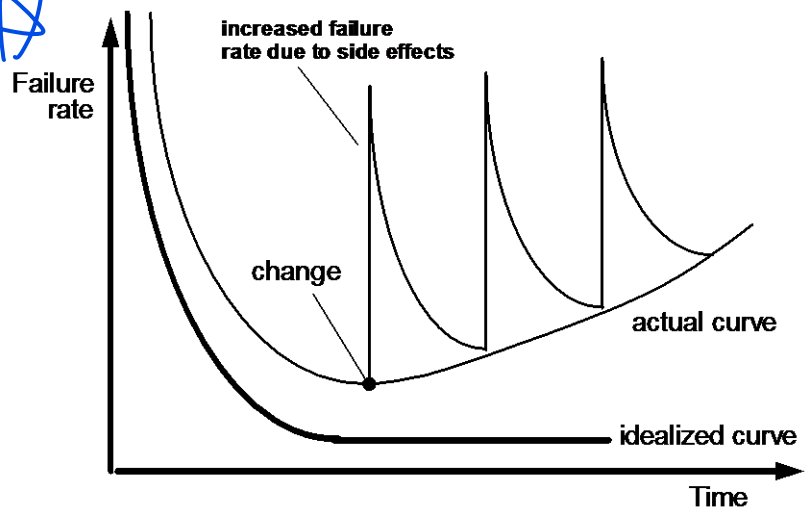
- ✓ Delivers Computing potential
- ✓ It resides on various resources
- ✓ Delivers imp product Time : Information

- ✓ Gateway to worldwide information
- ✓ Sophistication & complexity
- ✓ Adoption of software engg practice

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**



- 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.
- **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).
- 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
-

Software Engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

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 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 product classified in 2 classes:
1. Generic software: developed to solution whose requirements are very common fairly stable and well understood by software engineer.
2. Custom software: developed for a single customer according to their specification.

Need of Software Engineering: -
The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.
Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
Cost- As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
Quality Management- Better process of software development provides better and quality software product.

Software Engineering Goals

- Readability (understood by those who maintain it)
- Correctness
- Reliability (high performance)
- Re usability
- Extensibility (ability to perform its operations)
- Flexibility
- Efficiency

Software Crisis

- Many software projects failed.
- Many software projects late, over budget, providing unreliable software that is expensive to maintain.
- Many software projects produced software which did not satisfy the requirements of the customer.
- Complexities of software projects increased as hardware Capability increased.
- Larger software system is more difficult and expensive to maintain.
- Demand of new software increased faster than ability to generate new software.

Characteristics of good software: -

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

- Operational
- Transitional
- Maintenance
- Well-engineered and crafted software is expected to have the following characteristics:

Operational: -

This tells us how well software works in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

Transitional: -

This aspect is important when the software is moved from one platform to another:

- Portability
- Interoperability
- Reusability
- Adaptability

Maintenance: -

This aspect briefs about how well software has the capabilities to maintain itself in the ever-changing environment:

- Modularity
- Maintainability
- Flexibility
- Scalability

In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

SOFTWARE PROCESS MODEL:

Software process can be defining as the structured set of activates that are required to develop the software system.To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods, and tools layers. This strategy is

often referredto as a process model or a software engineering paradigm.
A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.
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.

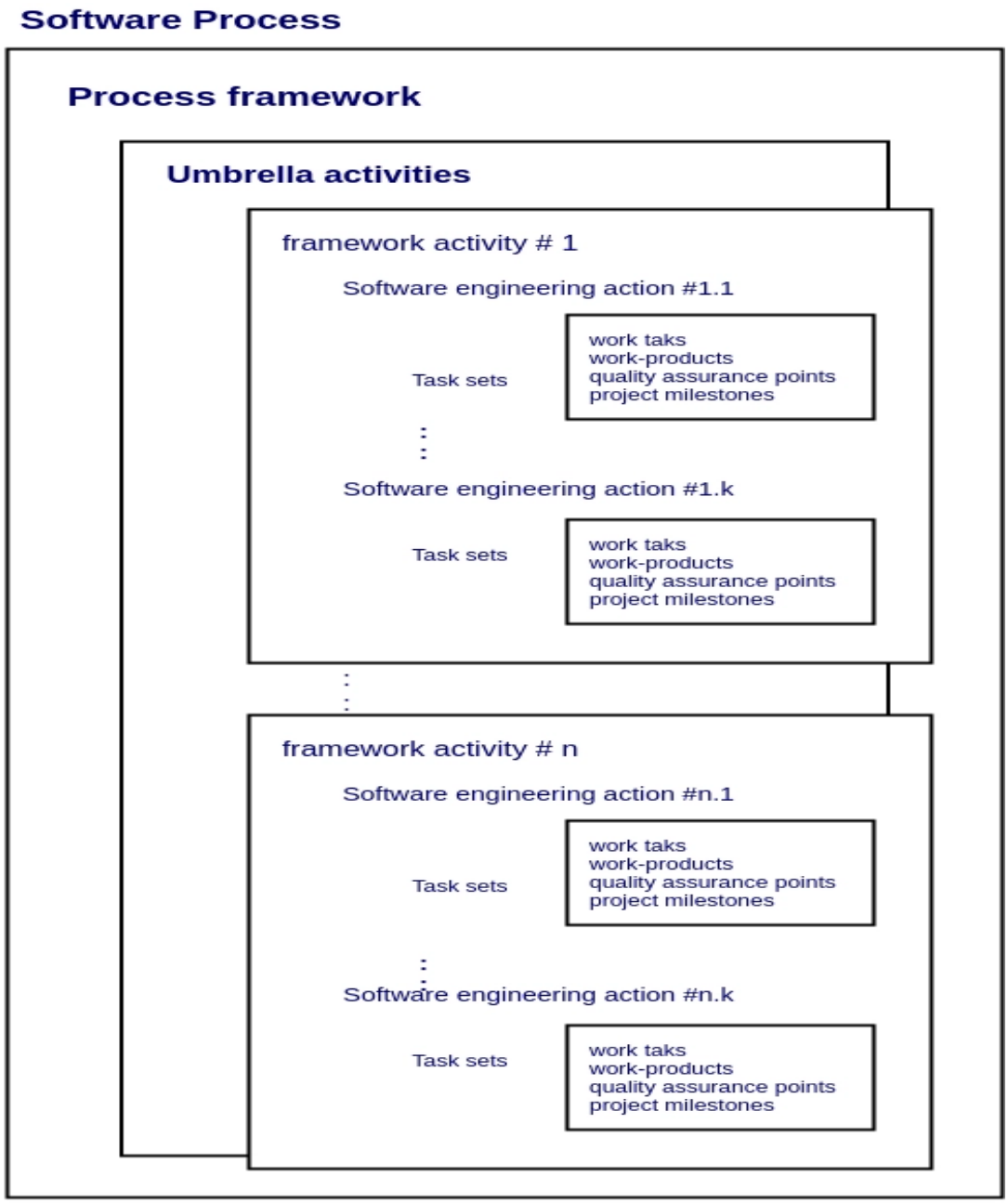


Fig. Common Software Process Framework

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.

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 Software Quality Assurance

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.
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.**

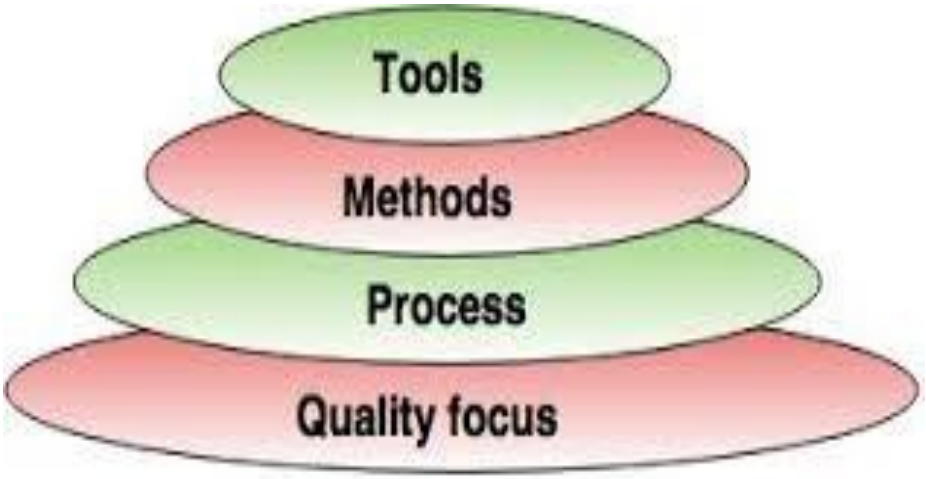


Fig. - Software Engineering Layers

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

The goal of a software process model is to provide guidance for systematically coordinating and controlling the tasks that must be performed in order to achieve the end product and their project objectives. A process model defines the following:

- A set of tasks that need to be performed.
- The inputs to and output from each task.
- The preconditions and post-conditions for each task.
- The sequence and flow of these tasks.

We might ask whether a software development process is necessary if there is only one person developing the software. The answer is that it depends. If the software development process is viewed as only a coordinating and controlling agent, then there is no need since there is only one person. However, if the process is viewed as prescriptive road map for generating various intermediate deliverables in addition to the executable code-for

Characteristics of Software Process: -

Software is often the single largest cost item in a computer-based application. Though software is a product, it is different from other physical products.

i. Software costs are concentrated in engineering (analysis and design) and not in production.
ii. Cost of software is not dependent on volume of production.
iii. Software does not wear out (in the physical sense).
iv. Software has no replacement (spare) parts.
v. Software maintenance is a difficult problem and is very different from hardware (physical product) maintenance.
vi. Most software is custom-built.
vii. Many legal issues are involved (e.g. inter-actual property rights, liability).

Software Product: -

A software product, user interface must be carefully designed and implemented because developers of that product and users of that product are totally different. In case of a program, very little documentation is expected, but a software product must be well documented. A program can be developed according to the programmer's individual style of development, but a software product must be developed using the accepted software engineering principles.

Various Operational Characteristics of software are:

- **Correctness:** The software which we are making should meet all the specifications stated by the customer.
- **Usability/Learn-ability:** The amount of efforts or time required to learn how to use the software should be less. This makes the software user-friendly even for IT-illiterate people.
- **Integrity:** Just like medicines have side-effects, in the same way software may have aside-effect i.e. it may affect the working of another application. But quality software should not have side effects.
- **Reliability:** The software product should not have any defects. Not only this, it shouldn't fail while execution.
- **Efficiency:** This characteristic relates to the way software uses the available resources. The software should make effective use of the storage space and execute command as per desired timing requirements.

- **Security:** With the increase in security threats nowadays, this factor is gaining importance. The software shouldn't have ill effects on data / hardware. Proper measures should be taken to keep data secure from external threats.
- **Safety:** The software should not be hazardous to the environment/life.

Difference between software process and software product: -

Software Process	Software Product
Processes are developed by individual user and it is used for personal use.	It is developed by multiple users and it is usedby large number of people or Customers.
Process may be small in size and possessing limited functionality.	It consists of multiple program codes; relate documents such as SRS, designing documents, user manuals, test cases.
Process is generally developed by process engineers.	Process is generally developed by process engineers. Therefore systematic approachof developing software product must be applied.
Software product relies on software process for its stability quality and control Only one person uses the process, hence lack of user interface	It is important than software product. Multiuser no lack of user interface.

Software Development Life Cycle/Process model/ Software Development Life Cycle: -

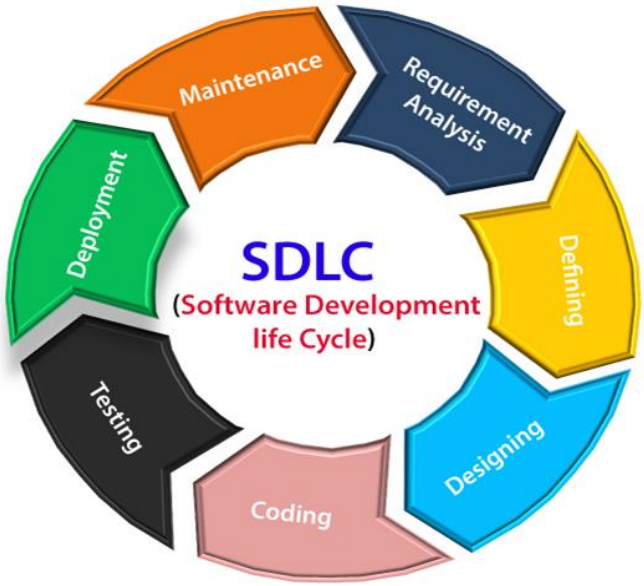
Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product. it is a team of engineers must incorporate a development strategy that encompasses the process, method and tools layers. Each phase has various activities to develop the software product. It also specifies the order in which each phase must be executed.
A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed.

Definition: Software Development Life Cycle (SDLC) is a process used by software industry to design, develop and test high quality software. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
SDLC is the acronym of Software Development Life Cycle. It is also called as Software development process. The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.

- 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)

- 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.
- 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)



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

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

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.

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

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.

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.

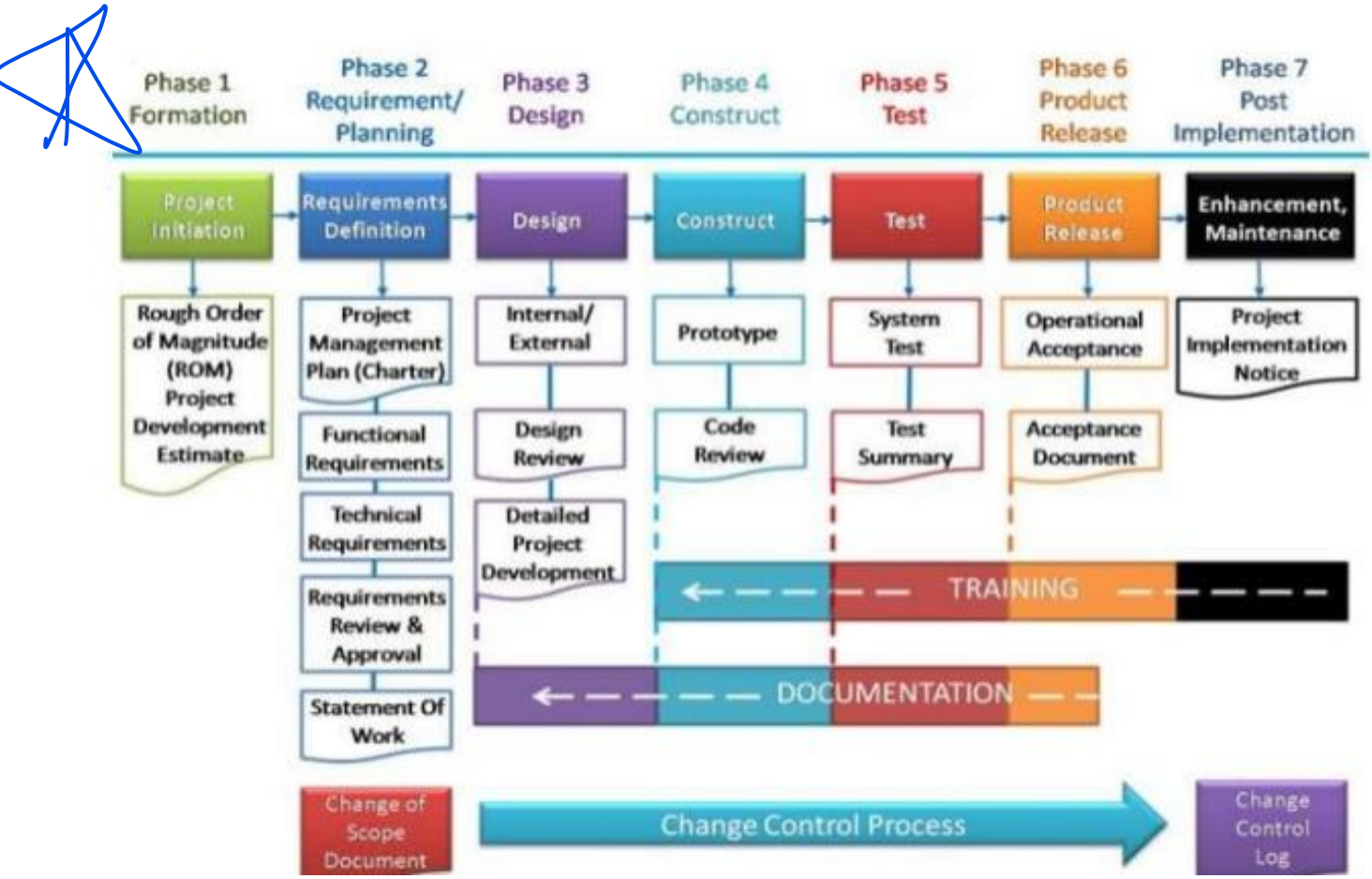
5. Testing the Software

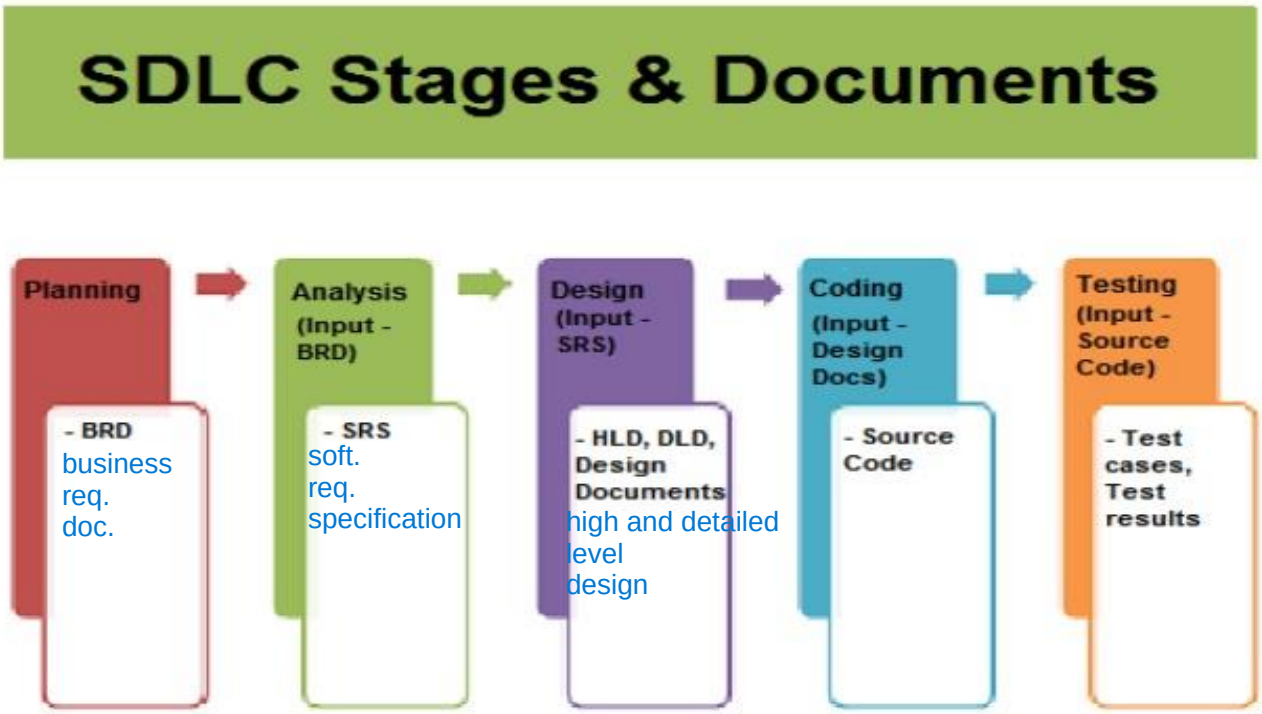
• This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. • However this stage refers to the testing only that stage of the software where defects are reported, tracked, fixed and retested, until the software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

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.





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.

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 Models

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

Prescriptive process model

- 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.
- 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.

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

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

-

LINEAR SEQUENTIAL MODEL:

A few of software development paradigms or process models are defined as follows:

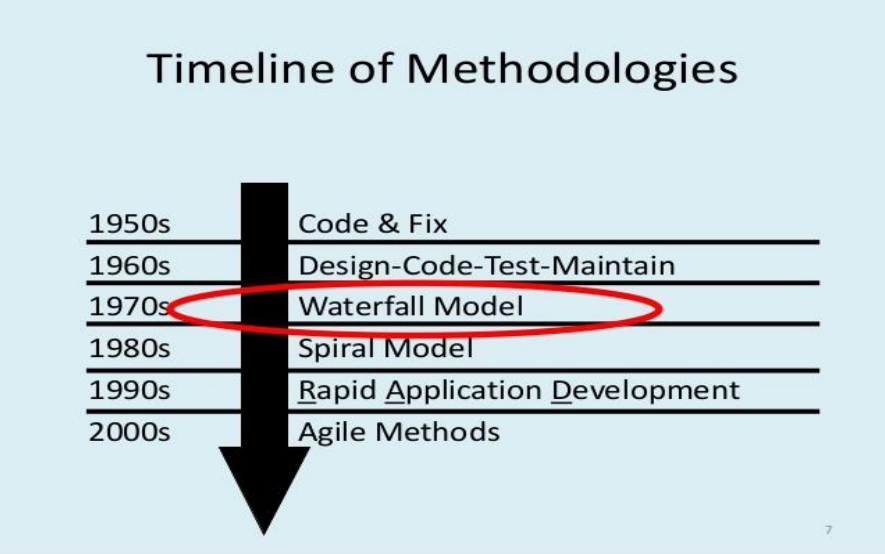
Waterfall model or linear sequential model or classic life cycle model: -

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.

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.

Sometimes called the classic life cycle or the waterfall model, the linear sequential model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and maintenance.



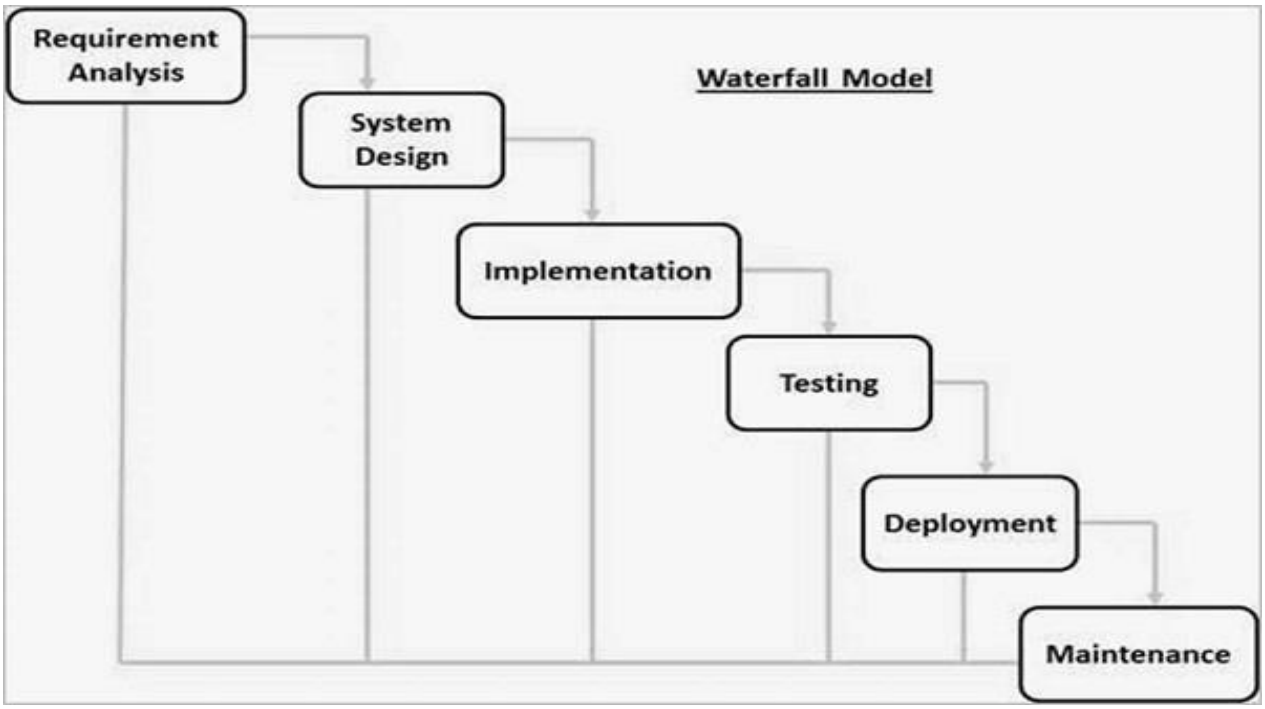


Figure 1.1: Waterfall model

- **Software requirements analysis**: The requirements gathering process is focused specifically on software. To understand the nature of the program(s) to be built, the software engineer ("analyst") must understand the information domain for the software, as well as required function, behavior, performance, and interface. Requirements for both the system and the software are documented and reviewed with the customer. 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.

Design: Software design is actually a multi-step process that focuses on four distinct attributes of a program: data structure, software architecture, interface representations, and procedural (algorithmic) detail. The design process translates requirements into a representation of the software that can be assessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.

- 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. Ø
- 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. Ø
- In short, this phase lays a fundamental for actual programming and implementation.
- 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.
-

Development / Coding : The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.

- 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.

Testing: Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree withrequired results. 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.

Deployment & Maintenance: Software will undoubtedly undergo change after it is delivered to the customer (a possible exception is embedded software). Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g., a change required because of a new operating system or peripheral device), or because the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one. 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. Ø

- 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.

Advantages of waterfall model: -

- This model is simple and easy to understand and use.
- Waterfall model works well for smaller projects where requirements are very well understood.
- Each phase proceeds sequentially.
- Documentation is produced at every stage of the software's development. This makes understandingthe product designing procedure, simpler.
- After every major stage of software coding, testing is done to check the correct running of the code. help us to control schedules and budgets.
- 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.

Disadvantages of waterfall model: -

- 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.
- High amounts of risk and uncertainty.
- Customer can see working model of the project only at the end. after reviewing of the working model if

the customer gets dissatisfied then it causes serious problem.

- You cannot go back a step if the design phase has gone wrong, things can get very complicated in theimplementation phase.
- 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

PROTOTYPING MODEL:

A prototype is a toy implementation of the system. A prototype usually exhibits limited functional capabilities, low reliability, and inefficient performance compared to the actual software. A prototype is usually built using several shortcuts. The shortcuts might involve using inefficient, inaccurate, or dummy functions. The shortcut implementation of a function, for example, may produce the desired results by using a table look-up instead of performing the actual computations. A prototype usually turns out to be a very crude version of the actual system.

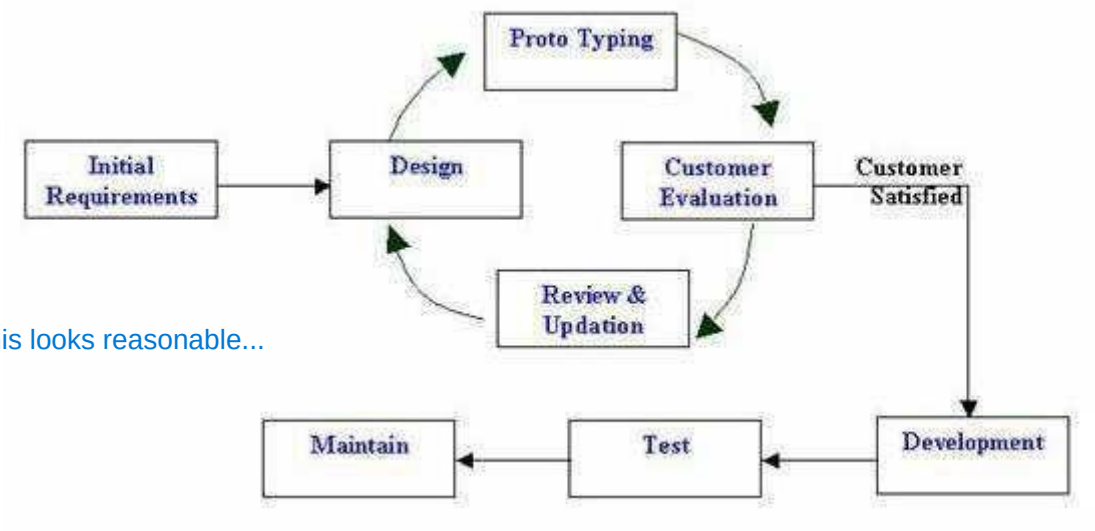


Figure 1.2: Prototype Model

Need for a prototype in software development: -

There are several uses of a prototype. An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer. This is a valuable mechanism for gaining better understanding of the customer's needs:

- how the screens might look like
- how the user interface would behave
- how the system would produce outputs

A prototyping model can be used when technical solutions are unclear to the development team.

A developed prototype can help engineers to critically examine the technical issues associated with the product development. Often, major design decisions depend on issues like the response time of a hardware controller, or the efficiency of a sorting algorithm, etc.

In such circumstances, a prototype may be the best or the only way to resolve the technical issues.

A prototype of the actual product is preferred in situations such as:

- user requirements are not complete
- technical issues are not clear

RAPID APPLICATION MODEL:

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product. In RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.

1. Since there is no detailed pre-planning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype. The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable. 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.
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.

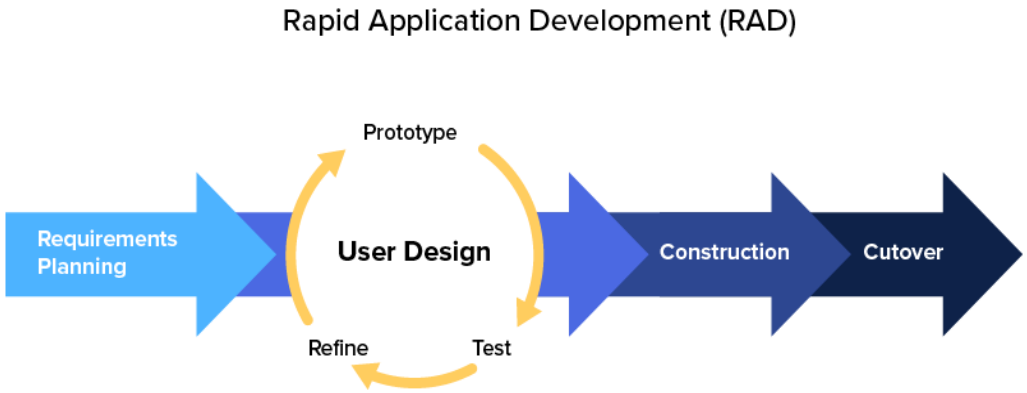
The main features of RAD model are that it focuses on the reuse of templates, tools, processes and code.

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.



Rapid application development (RAD) is an incremental software development process model that emphasizesan extremely short development cycle. The ‘AD model is a “high-speed” adaptation of the linear sequential model in which rapid development is achieved by using component-based construction. If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within very short time periods (e.g., 60 to 90 days). Used primarily for information systems applications, **the RAD approach encompasses the following phases:**

- **Business modeling:** The information flow among business functions is modeled in a way that answers the following questions: What information drives the business process? What information is generated? Who generates it? Where does the information go? Who processes it? **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

- **Data modeling:** The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business. The characteristics (called *attributes*) of each object are identified and the relationships between these objects defined. **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.

The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

- **Application generation:** RAD assumes the use of fourth generation techniques. Rather than creating software using conventional third generation programming languages the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary). In all cases, automated tools are used to facilitate construction of the software. 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.

Since the RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time. However, new components must be tested and all interfaces must be fully exercised.

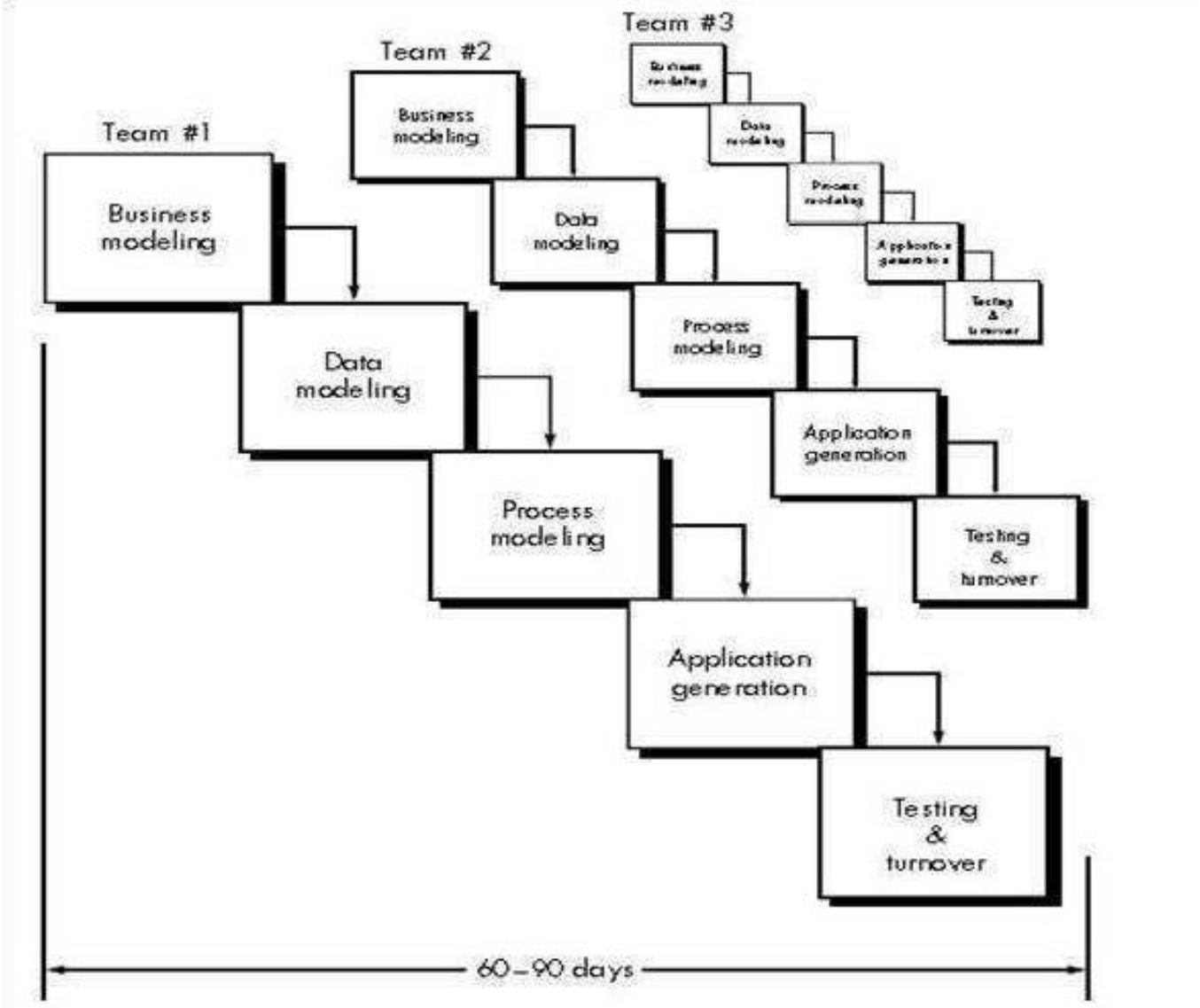


Figure 1.3: RAPID APPLICATION MODEL

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.

Advantages of the RAD model:

- Reduced development time.
- Increases re usability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.
- 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.
-

Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.
- 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.
-

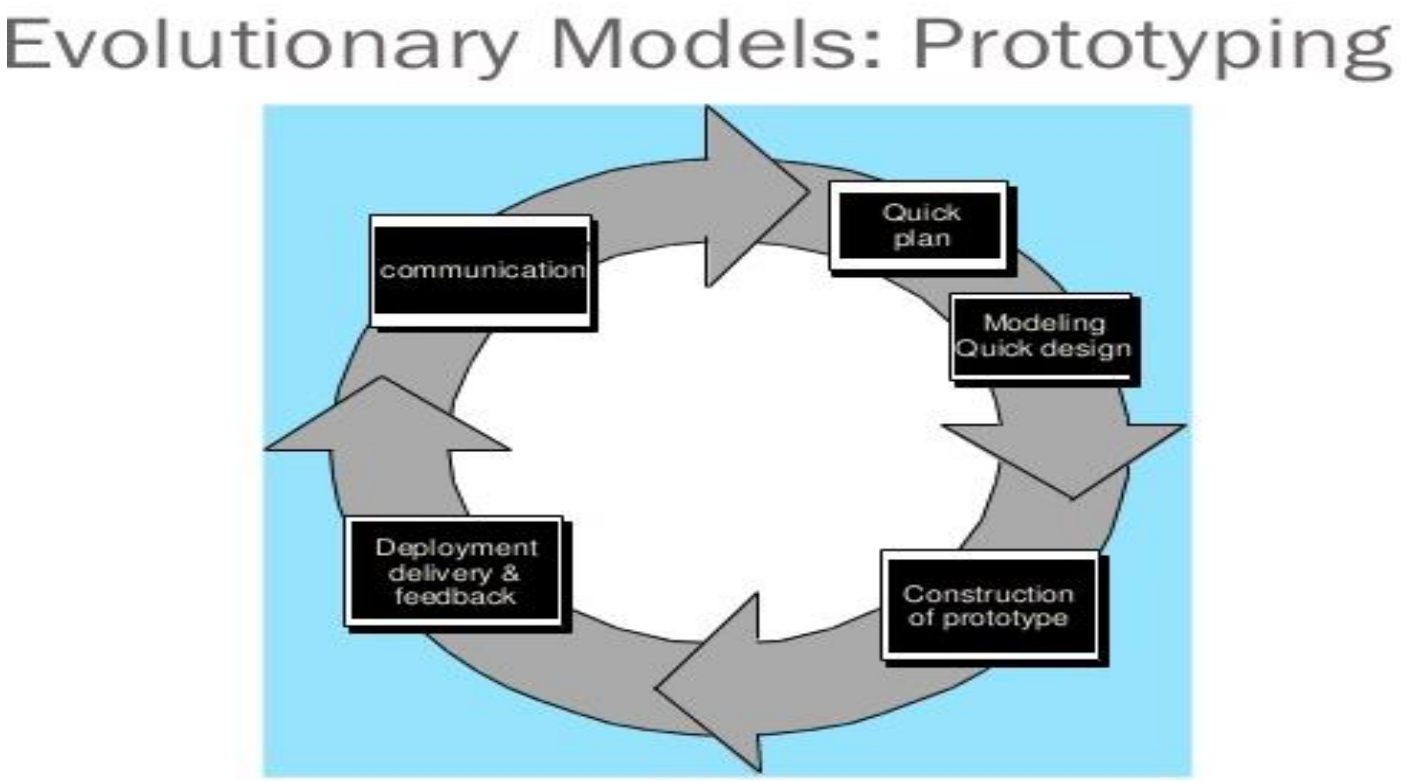
When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in 2-3 monthsof 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 andthere is a need to produce the system in a short span

of time (2-3 months).

EVOLUTIONARY PROCESS MODEL:

Evolutionary Software Process Model Evolutionary software models are iterative. They are characterized in manner that enables the software engineers to develop increasingly more complete version of software. In programming "iteration" means sequential access to objects. It is typically a cycle. Software engineers can follow this process model that has been clearly designed to put up a product that regularly complete over time.



Iterative Model design:

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Following is the pictorial representation of Iterative and Incremental model:

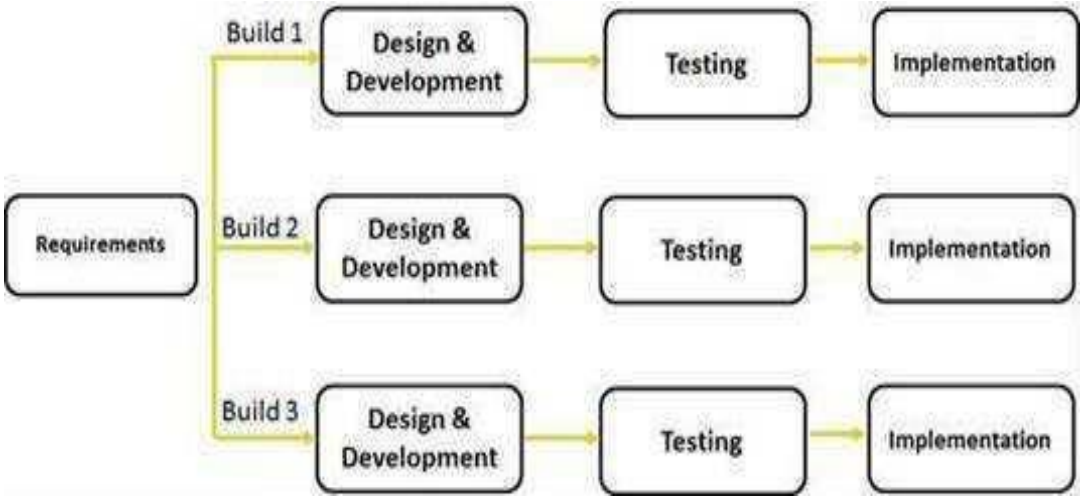


Figure 1.4: Iterative Model

Iterative Model Application:

Like other SDLC models, Iterative and incremental development has some specific applications in the softwareindustry. This model is most often used in the following scenarios:

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements mayevolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on theproject.
- Resources with needed skill set are not available and are planned to be used on contract basis forspecific iterations.
- There are some high-risk features and goals which may change in the future.

Evolutionary Process Model is of 2 types

- incremental model and
- spiral model

INCREMENTAL MODEL:

- The incremental model combines the elements of waterfall model and they are applied in an iterativefashion.
- The first increment in this model is generally a core product.
- Each increment builds the product and submits it to the customer for any suggested modifications.
- The next increment implements on the customer's suggestions and add additional requirements in theprevious increment.
- This process is repeated until the product is finished.
- ● 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.

For example, the word-processing software is developed using the incremental model.

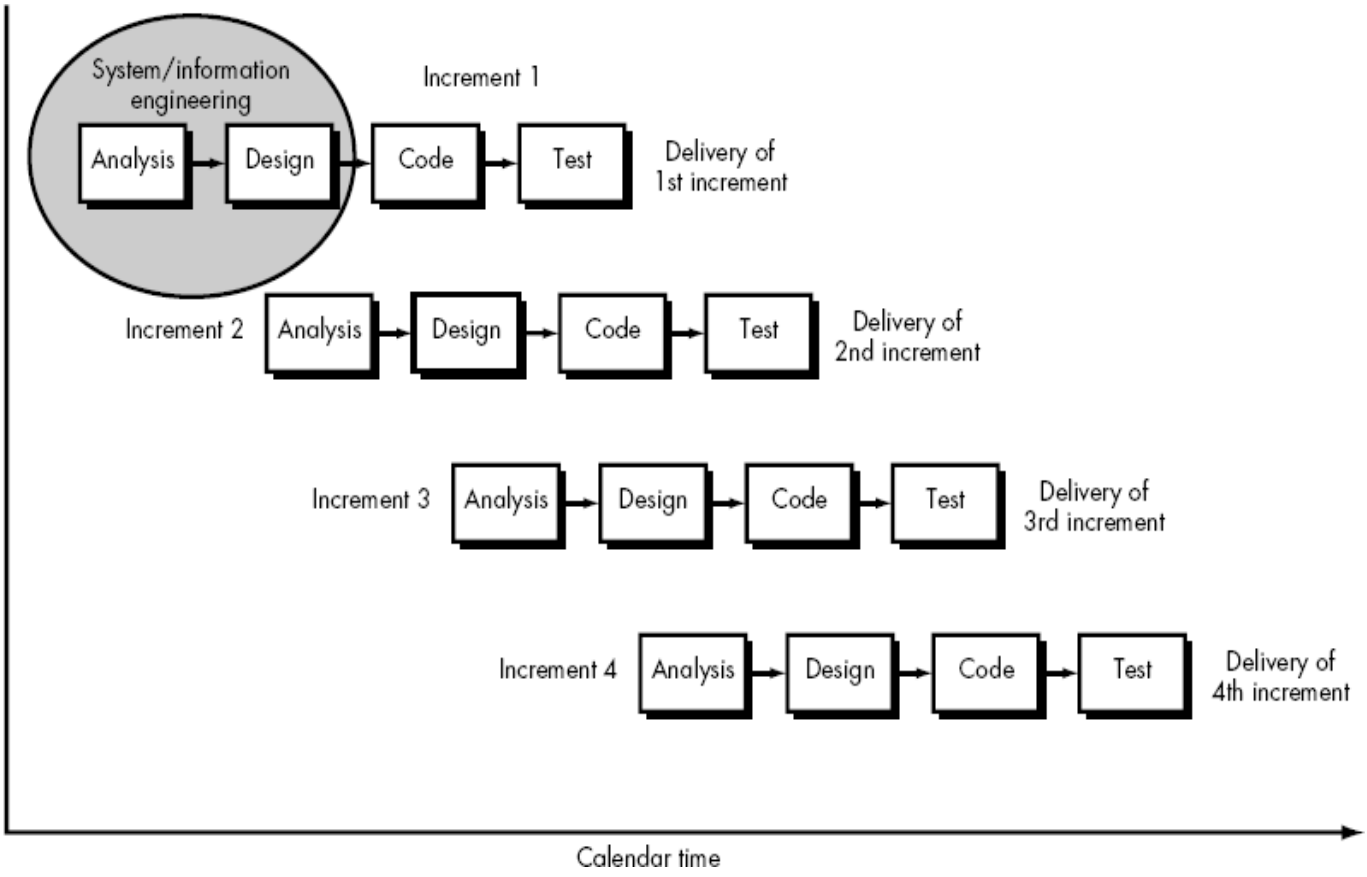


Figure 1.5: Incremental Model

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'd iteration.
- There is low risk for overall project failure.
- Customer does not have to wait until the entire system is delivered.

Disadvantages of Incremental model: -

- Needs good planning and design at the management a technical level.
- Needs a clear and complete definition of the whole system before it can be broken

down and builtincrementally.
- Total cost is higher than waterfall.
- Time foundation create problem to complete the project.

When to use the Incremental model:

- This model can be used when the requirements of the complete system are clearly defined andunderstood.
- 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 goals.

SPIRAL MODEL :

The spiral model, is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. It provides the potential for rapid development of incremental versions of the software. Using the spiral model, software is developed in a seriesof incremental releases. During early iterations, the incremental release might be a paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

- 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.
- The spiral model was defined by Barry Boehm in his 1988 article. v
- This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters.
  Perfomed on each phase:-
- **Concept developmen project**
- **New product development projects**
- **product inhancement projects**
- **product maintenance projects**
-

A spiral model is divided into a number of framework activities, also called task regions. Project entry point axisis defined this axis represents starting point for different types of project. Every framework activities represent one section of the spiral path. As the development process starts, the software team performs activities that are indirect by a path around the spiral model in a clockwise direction. It begins at the center of spiral model.
Typically, there are between three and six task regions. In blow Figure depicts a spiral model that contains sixtask regions:

- **Customer communication**—tasks required to establish effective communication between developer andcustomer.
- **Planning**—tasks required to define resources, time lines, and other project related information.
- **Risk analysis**—tasks required to assess both technical and management risks.
- **Engineering**—tasks required to build one or more representations of the application.
- **Construction and release**—tasks required to construct, test, install, and provide user

support(e.g.,documentation and training).

- **Customer evaluation**—tasks required to obtain customer feedback based on evaluation of the softwarerepresentations created during the engineering stage and implemented during the installation stage.

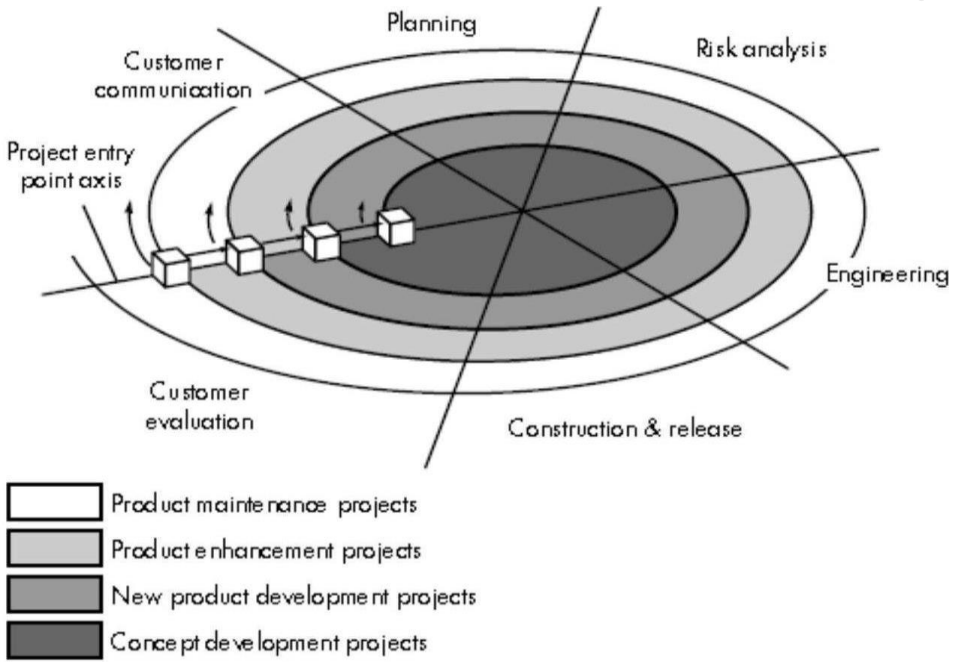


Figure 1.6: Spiral Model

Advantages of Spiral model: -

- 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.
- Software is produced early in the software life cycle.

Disadvantages of Spiral model: -

- 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.

When to use Spiral model:

- When costs and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment unwise because of potential changes to economic

priorities.

- Users are unsure of their needs.
- Requirements are complex.
- New product line.
- Significant changes are expected (research and exploration).

- # 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

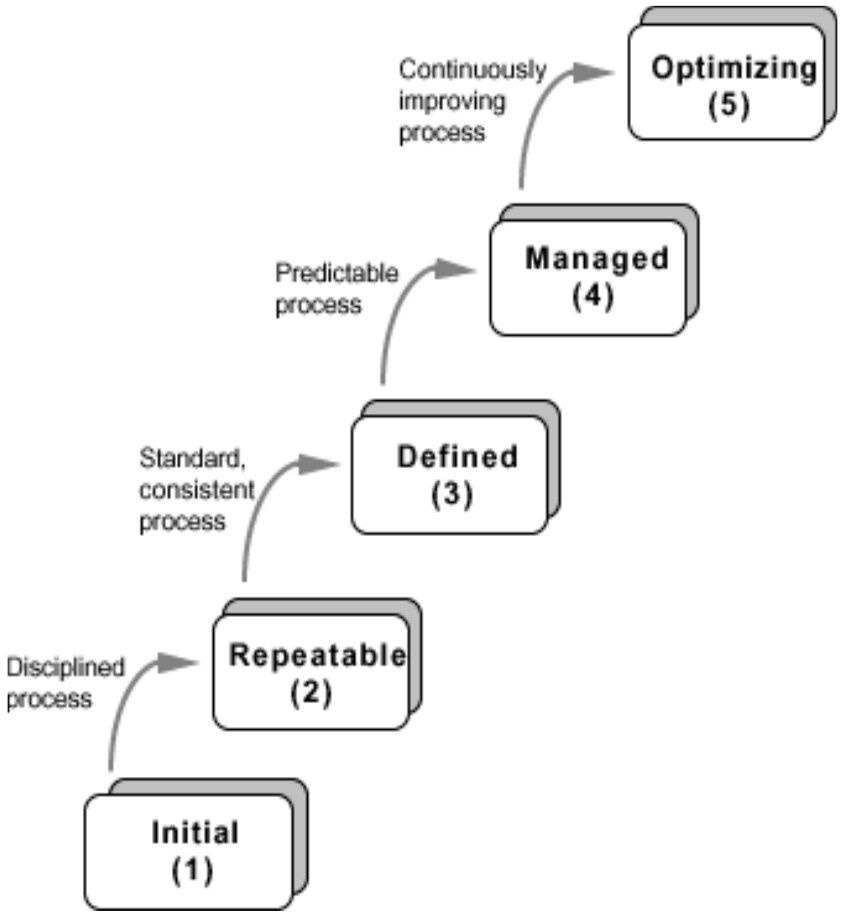
CMM: Capability Maturity Model

- Developed by the Software Engineering Institute of the Carnegie Mellon University
- Framework that describes the key elements of an effective software process.
- 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.
- **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 five levels of software process** maturity
- Maturity level indicates level of process capability:
- Initial
- Repeatable
- Defined
- Managed

- Optimizing
-



CAPABILITY MATURITY MODEL (CMM):

The Software Engineering Institute (SEI) has developed a comprehensive model predicated on a set of softwareengineering capabilities that should be present as organizations reach different levels of process maturity. To determine an organization's current state of process maturity, the SEI uses an assessment that results in a five-point grading scheme. The grading scheme determines compliancewith a capability maturity model (CMM) that defines key activities required at different levels of process maturity. The SEI approach provides a measure of the global effectiveness of a company's software engineering practices and establishes five process maturity levels that are defined in the following manner:

Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort. 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 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3: Defined. The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2.

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 using detailed measures. This level includes all characteristics defined for level 3. 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 testing innovative ideas and technologies. This level includes all characteristics defined for level 4.

- 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

The five levels defined by the SEI were derived as a consequence of evaluating responses to the SEI assessmentquestionnaire that is based on the CMM. The results of the questionnaire are distilled to a single numerical grade that provides an indication of an organization's process maturity.

Figure 1.9: Capability Maturity Model

➨ 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

Process maturity level 2: -

- Software configuration management
- Software quality assurance
- Software subcontract management
- Software project tracking and oversight
- Software project planning
- Requirements management

Process maturity level 3: -

- Peer reviews
- Intergroup coordination

- Software product engineering
- Integrated software management
- Training program
- Organization process definition
- Organization process focus

Process maturity level 4: -

- Software quality management
- Quantitative process management

Process maturity level 5: -

- Process change management
- Technology change management
- Defect prevention

RATIONAL UNIFIED PROCESS (RUP):

Rational Unified Process (RUP) is an object-oriented and Web-enabled program development methodology. RUP is a software application development technique with many tools to assist in coding the final product and tasks related to this goal. RUP is an object-oriented approach used to ensure effective project management and high-quality software production. It divides the development process into four distinct phases that each involves business modeling, analysis and design, implementation, testing, and deployment. The four phases are:

1. **Inception** - The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.
   - Communication and planning are main.
   - Identifies Scope of the project using use-case model allowing managers to estimate costs and time required.
   - Customers requirements are identified and then it becomes easy to make a plan of the project.
   - Project plan, Project goal, risks, use-case model, Project description, are made.
   - 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** - The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.
   - Planning and modeling are main.

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

3. **Construction** - The project is developed and completed. The software is designed, written, and tested.

   - Project is developed and completed.
   - System or source code is created and then testing is done.
   - 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** - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

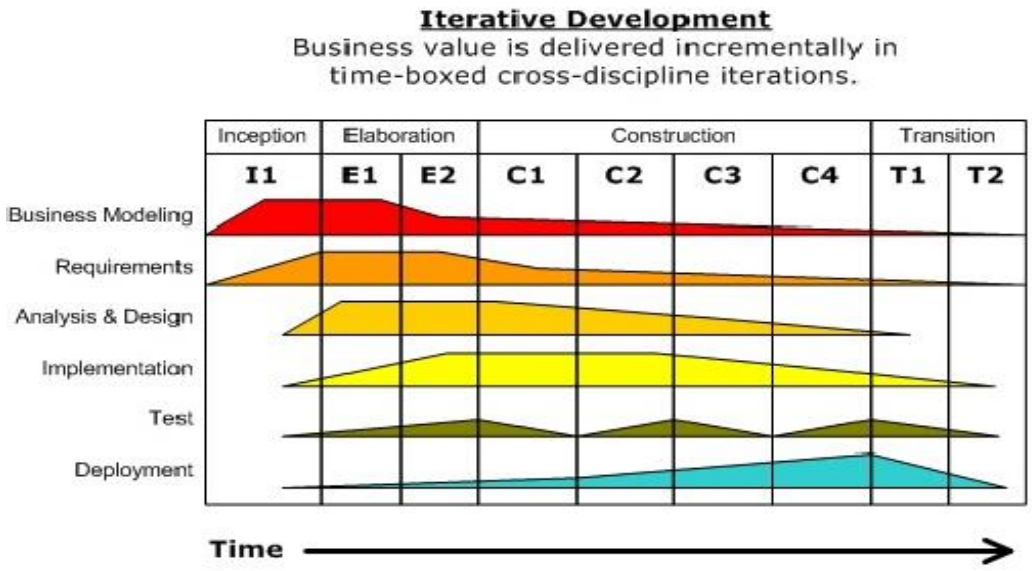
   - Final project is released to public.
   - Transit the project from development into production.
   - Update project documentation.
   - Beta testing is conducted.
   - Defects are removed from project based on feedback from public.

   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 –**
   - Final phase of the model.
   - Project is maintained and updated accordingly.



The RUP development methodology provides a structured way for companies to envision create software programs. Since it provides a specific plan for each step of the development process, it helps prevent resourcesfrom being wasted and reduces unexpected development costs.

Discipline of RUP

1. Business Modeling

Business modeling explains how to describe a vision of the organization in which the system will be deployed and how to then use this vision as a basis to outline the process, roles and responsibilities.

Organizations are becoming more dependent on IT systems, making it imperative that information system engineers know how the applications they are developing fit into the organization. Businesses invest in IT when they understand the competitive advantage and value added by the technology. The aim of business modeling is to first establish a better understanding and communication channel between business engineering and software engineering. Understanding the business means that software engineers must understand the structure and the dynamics of the target organization (the client), the current problems in the organization, and possible improvements. They must also ensure a common understanding of the target organization between customers, end users and developers.

2. Requirements

Requirements explain how to elicit stakeholder requests and transform them into a set of requirements work products that scope the system to be built and provide detailed requirements for what the system must do.

3. Analysis and Design

The goal of analysis and design is to show how the system will be realized. The aim is to build a system that:

- Performs — in a specific implementation environment — the tasks and functions specified in the use-case descriptions.
- Fulfills all its requirements.
- Is easy to change when functional requirements change.

Designs results in a design model and analysis optionally into an analysis model. The design model serves as an abstraction of the source code; that is, the design model acts as a 'blueprint' of how the source code is structured and written. The design model consists of design classes structured into packages and subsystems with well-defined interfaces, representing what will become components in the implementation. It also contains descriptions of how objects of these design classes collaborate to perform use cases.

4. Implementation

The purposes of implementation are:

- To define the organization of the code in terms of implementation subsystems that are organized in layers.
- To implement classes and objects in terms of components (source files, binaries, executables, and others).
- To test the developed components as units.
- To integrate the results produced by individual implementers (or teams) into an executable system.

Systems are realized through the implementation of components. The process describes how to reuse existing components, or implement new components with well-defined responsibility, making the system easier to maintain and increasing the possibilities to reuse.

5. Test

Test workflow: The purpose of testing is to assess product quality. This not only involves the final product, but also begins early in the project with the assessment of the architecture and continues through the assessment of the final product delivered to customers. The test workflow involves the following:

- Verifying the interactions of components
- Verifying the proper integration of components
- Verifying that all requirements have been implemented correctly
- Identifying and ensuring that all discovered defects are addressed before the software is deployed

6. Deployment

The purpose of deployment is to successfully produce product releases, and to deliver the software to its end users. It covers a wide range of activities including producing external releases of the software, packaging the software and business application, distributing the software, installing the software, and providing help and assistance to users. Although deployment activities are mostly centered around the transition phase, many of the activities need to be included in earlier phases to prepare for deployment at the end of the construction phase. The Deployment and Environment workflows of the Rational Unified Process contain less detail than other workflows.

Advantages of RUP Software Development: -

- This is a complete methodology in itself with an emphasis on accurate documentation
- It is pro-actively able to resolve the project risks associated with the client's evolving requirementsrequiring careful change request management
- Less time is required for integration as the process of integration goes on throughout the softwaredevelopment life cycle.
- The development time required is less due to reuse of components.
- There is online training and tutorial available for this process.

Disadvantages of RUP Software Development: -

- The team members need to be expert in their field to develop a software under this methodology.
- The development process is too complex and disorganized.
- On cutting edge projects which utilize new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfill.

- Integration throughout the process of software development, in theory sounds a good thing. But on particularly big projects with multiple development streams it will only add to the confusion and cause more issues during the stages of testing.

Figure 1.8: Rational

Unified Process (RUP

SOFTWARE PROCESS CUSTOMIZATION:

In software industry, most of the projects are customized software product 3 major factors that are involved insoftware process customization and those are:

- PEOPLE
- PRODUCT
- PROCESS

People: -

The primary element of any project is the people. People gather requirements, people interview users (people), people design software, and people write software for people. No people -- no software. I'll leave thediscussion of people to the other articles in this special issue, except for one comment. The best thing that canhappen to any software project is to have people who know what they are doing and have the courage and self-discipline to do it. Knowledgeable people do what is right and avoid what is wrong. Courageous people tell the truth when others want to hear something else. Disciplined people work through projects and don't cut corners. Find people who know the product and can work in the process.

Process: -

Process is how we go from the beginning to the end of a project. All projects use a process. Many project managers, however, do not choose a process based on the people and product at hand. They simply use the same process they've always used or misused. Let's focus on two points regarding process: (1) process improvement and (2) using the right process for the people and product at hand.

Product: -

The product is the result of a project. The desired product satisfies the customers and keeps them coming backfor more. Sometimes, however, the actual product is something less. The product pays the bills and ultimately allows people to work together in a process and build software. Always keep the product in focus.

PRODUCT AND PROCESS METRICS:

Software process metrics measure the software development process and environment. Example productivity,effort estimates, efficiency and failure rate. Software Product metrics measure the software product.Example: - size, reliability, complexity and functionality.

Process Metrics and Software Process Improvement: -

The only rational way to improve any process is

- To measure specific attributes of the process
- Develop a set of meaningful metrics based on these attributes
- Use the metrics to provide indicators that will lead to a strategy for improvemT.

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 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.
- 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.

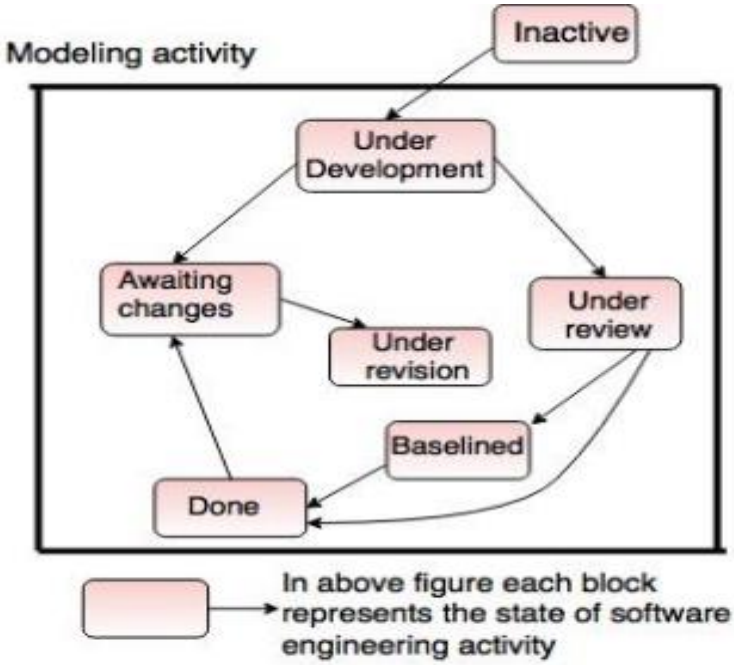


Fig. - One element of the concurrent process model

- **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.

- **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.

- 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

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.

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

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.

- 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)

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.

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.

- ***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 Software Process (PSP)**

The PSP model defines five framework activities

✓ Planning
✓ High-level design
✓ High-level design review

- ✓ Development
- ✓ Postmortem
-

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.

2. Team Software Process (TSP)

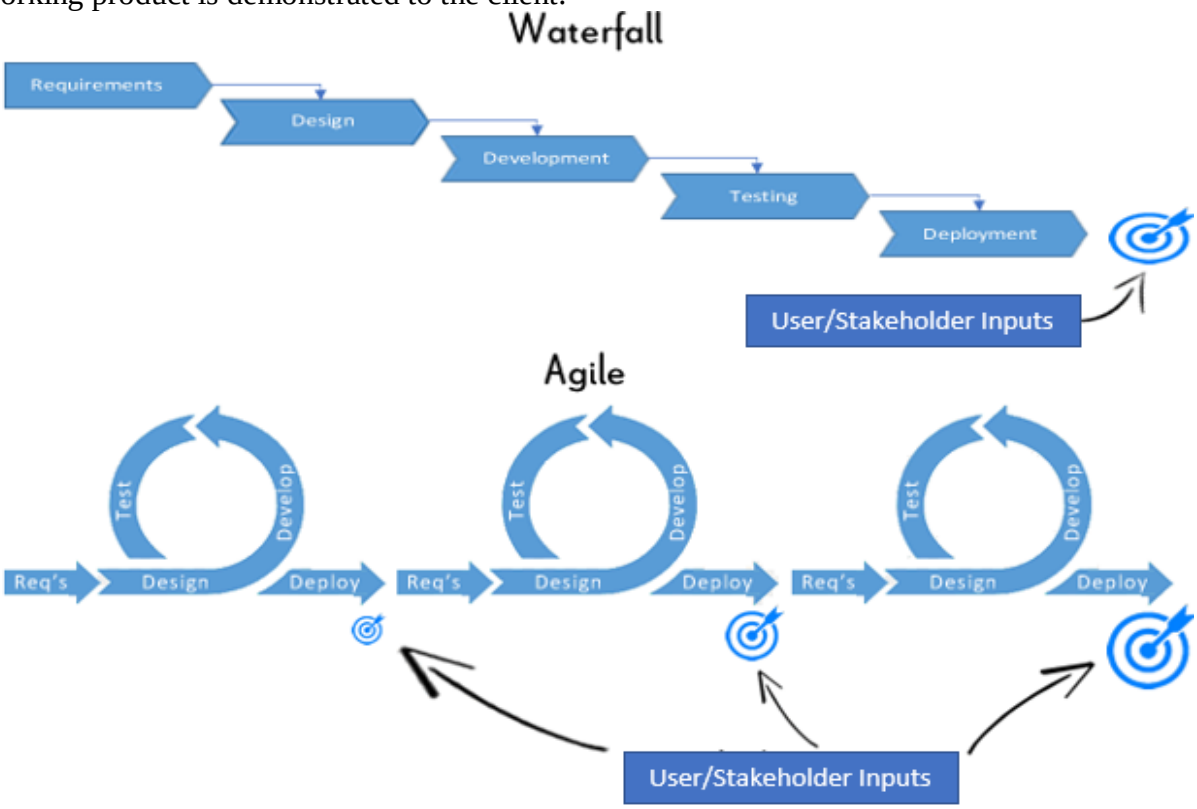
Watts Humphrey extended the lessons learned from the introduction of PSP and proposed a Team Software Process (TSP). The goal of TSP is to build a "self directed" project team that organizes itself to produce high quality software.

- ❑ **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.
- ❖ 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.
- 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.



The four core values outlined in the Agile Manifesto are:

Individual interactions are more important than processes and tools. People drive the development process and respond to business needs. They are the most important part of development and should be valued above processes and tools. If the processes or tools drive development, then the team will be less likely to respond and adapt to change and, therefore, less likely to meet customer needs.

A focus on working software rather than thorough documentation. Before Agile, a large amount of time was spent on documenting the product throughout development for delivery. The list of documented requirements was lengthy and would cause long delays in the development process. While Agile does not eliminate the use of documentation, it streamlines it in a way that provides the developer with only the information that is needed to do the work

-- such as user stories. The Agile Manifesto continues to place value on the process of documentation, but it places higher value on working software.

Collaboration instead of contract negotiations. Agile focuses on collaboration between the customer and project manager, rather than negotiations between the two, to work out the details of delivery. Collaborating with the customer means that they are included throughout the entire development process, not just at the beginning and end, thus making it easier for teams to meet the needs of their customers. For example, in Agile software development, the customer may be included at different intervals for demos of the product. However, the customer could also be present and interacting with the teams on a daily basis, attending all meetings and ensuring the product meets their desires.

A focus on responding to change. Traditional software development used to avoid change because it was considered an undesired expense. Agile eliminates this idea. The short iterations in the Agile cycle allow changes to easily be made, helping the team modify the process to best fit their needs rather than the other way around. Overall, Agile software development believes change is always a way to improve the project and provide additional value.

The 12 principles of Agile

The Agile Manifesto also outlined 12 core principles for the development process. They are:

1. Satisfy customers through early and continuous delivery of valuable work.
2. Break big work down into smaller tasks that can be completed quickly.
3. Recognize that the best work emerges from self-organized teams.
4. Provide motivated individuals with the environment and support they need and trust them to get the job done.
5. Create processes that promote sustainable efforts.
6. Maintain a constant pace for completed work.
7. Welcome changing requirements, even late in a project.
8. Assemble the project team and business owners on a daily basis throughout the project.
9. Have the team reflect at regular intervals on how to become more effective, then tune and adjust behavior accordingly.
10. Measure progress by the amount of completed work.
11. Continually seek excellence.
12. Harness change for a competitive advantage.

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.

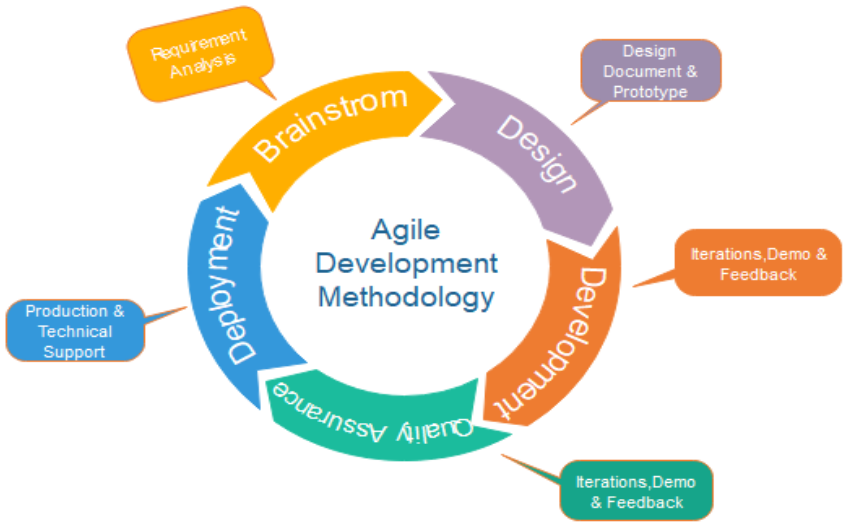


Fig. Agile Model

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.

Advantage(Pros) of Agile Model

5. Frequent Delivery
6. Face-to-Face Communication with clients.
7. Efficient design and fulfils the business requirement.
8. Anytime changes are acceptable.
9. It reduces total development time.

Disadvantage(cons) 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.