UNIT I
Introduction to Software Engineering

Introduction to Software Engineering:

Software crisis started in the mid of late 1960s and the early 1970s. The fields of computing have become complex and diverse in the modern information society. The main focus of practitioners from the computing outset was to provide solutions to the complexity barriers of producing software, escalating software industry and the number of software users. Thus, software came into existence. Nowadays, every small and large scale software development company is following the engineering disciplines for developing software.

The importance of software is developed due to several reasons:

1. The dependency of business organizations on software and technology has increased. Small and large scale organizations have automated their business processes for increased ease and effectiveness.
2. The varying nature and skills of users and customers have widely increased on information society. It is observed that most of the projects were unsuccessful due to unclear requirements provided from the customer.
3. The dynamic nature of software technology forces software companies to move towards component-based development, where components are assembled rather than developed from scratch.
4. Software allows changes in requirements at any stage in the development process but at maintenance stage it becomes tedious.

The ultimate goal of Software practitioners is to produce faster, better and cost-effective products. Hence, systematic approaches are needed for the development, management, retirement, and improvement.

Software:

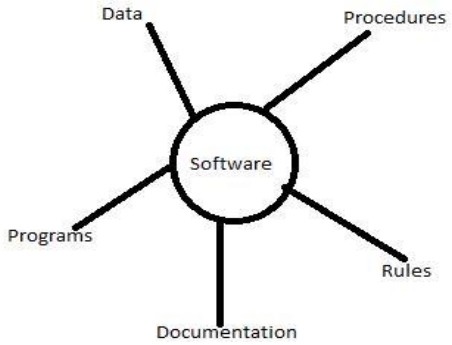
Definition:

It is a collection of computer programs that when executed together with data provide desired outcomes.

IEEE Definition:

It is a collection of computer programs, together with data, procedure, rules, and associated documentation, which operate in a specified environment with certain constraints to provide the desired outcomes.

Software View



- Data are the collection of facts, measurements, and statistics. Usually data are static in nature and can exist in any form, usable or not.
- Procedures allow automating process steps, ensuring compliance, governance, and repeatability.
- Rules are the guidelines for development, maintenance, and retirement of software.

- Documentation is important in understanding the software code, design, constraints, customer needs, and specification for further maintenance.

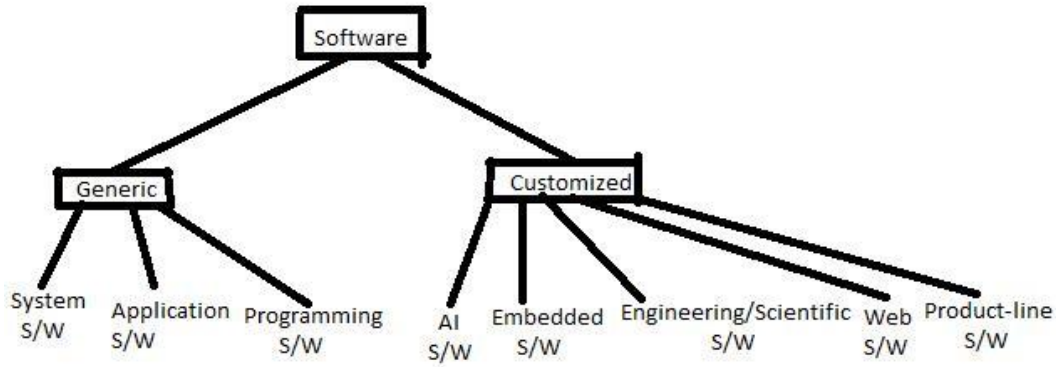
Characteristics of Software:

1. Software has logical properties rather than physical:
    i) Software has no physical shape, no volume, no colour, and no odour but s/w products can be measured, estimated and their performance can be calculated.
    ii) It logically consists of several programs connected through interfaces which are again programs that can be managed and implemented through programming languages.
    iii) The entire s/w product follows "divide and conquer" policy for its design and implementation.
2. S/W is produced in an engineering manner rather than in a classical manner:
    i) The engineering mechanism provides some organized activities or tasks with their defined approaches for s/w production.
    ii) Some general activities are feasibility study, analysis, design, coding, testing, and deployment.
    iii) It is not possible to produce s/w in a manufacturing manner because this would be a more complex process to manage and develop software.
3. S/W is mobile to change:
    i) S/W is too much flexible product that it can easily be changed. These changes may arise due to the changing requirements of the user and technological advancements.
    ii) Functionality can be modified, platform can be changed, new features can be incorporated, and software can further be migrated onto different platform. Due to changes, though quality remains the same, the products become very complex.
4. S/W becomes obsolete but does not wear out or die:
    i) S/W becomes obsolete due to increasing requirements of the users and rapidly changing technologies and do not wear out as they do not have any physical properties. H/W products can wear out due to environmental maladies and high failure rate.
    ii) The detection and correction of defects is a very difficult process in S/W, which causes S/W to become very costly.
    iii) S/W does not die but it can be made to retire after reengineering of the existing S/W milestones and the product becomes alive again.
5. S/W has certain operating environment, end user, and customer:
    i) Some s/w products are platform independent while others are platform specific. They are 3 levels of end users, i.e., top-level, middle-level and lower-level employees in an organization.
    ii) Some s/w products are user specific like games, embedded M/C's and so on.
6. S/W development is a labour-intensive task:
    i) Requirements are collected from the users and customers.
    ii) The s/w is designed and finally implemented in some programming language and on some platform, quality is tested and reused parts are verified and so on which follows all engineering steps.
    iii) Therefore, s/w is costlier when compared with H/W.

S/W Classifications:

- S/W can either be generic and customized.
- Generic and Customized S/W products can again be divided into several categories depending upon the type of customer, business, technology, and customer support.

S/W Classification



1. **System S/W:**

   It is a computer S/W which executes programs, transfers data between devices and controls, and operates the computer H/W.

   Ex: Device Drivers, boot program, operating systems, servers, utilities, and so on. System S/W reduces the burden of Application programmers.

2. **Application S/W:**

   It is designed to accomplish certain specific needs of the end user, by using the capabilities of the system S/W.

   Ex: Enterprise s/w, sale transaction s/w, Educational s/w, video editing s/w, word processing, spreadsheet etc.

   These may be available in different forms such as licensed, sold, freeware, shareware, open source, need to be installed, run online etc.

3. **Programming S/W:**

   Is a class of system s/w which helps programmers in writing computer programs using different programming languages in a convenient manner.

   Ex: Text Editors, Compilers, Interpreters, debuggers, linkers, and loaders etc, all in the form of a single application called IDE-Integrated Development Environment.

4. **AI S/W:**

   It is made to think like human beings and therefore it is useful in solving complex problems automatically. It uses techniques or algorithms for working programs to represent and manipulate knowledge.

   Ex: Game Playing, Speech recognition, understanding natural language, robotics etc are some AI applications.

5. Embedded S/W:

Type of s/w that is built in h/w systems. It is used to control, monitor or assist the operation of equipment, machinery, or plant.

Ex: 1. Applications or Daily Life Examples- Washing M/C's, cars, mobiles etc.

2. Controllers, real-time OS's, communication protocols are examples of Embedded S/W.

6. Engineering/Scientific S/W:

Engineering problems and quantitative analysis are carried out using automated tools. Scientific S/W is typically used to solve mathematical functions and calculations. Computer-Aided Design and Computer Aided Manufacturing S/W (CAD/CAM), civil engineering etc.

7. Web S/W:

These have been evolved from a simple website to search engines to web computing, which are spread over the N/W.

Web applications are based on client-server architecture, where the client requests information and the server stores and retrieves information from the web. Ex: Web 2.0, HTML, PHP, search engines

8. Product-Line S/W:

It is a set of S/W systems that a common, managed set of features to satisfy the specific needs of a particular market segment or mission. These are developed from a common set of core asserts in a prescribed way. Product line s/w improves time to market, productivity, quality and other business drivers. Applications: Multimedia, database s/w, word processing etc.

Engineering Discipline:

It is a disciplined approach with some organized steps in a managed way to construction, operation & maintenance of the s/w.

Early s/w practitioners followed an exploratory programming pattern in which there was an adhoc development of s/w.

Each programmer followed his own style of program development.

As the development of s/w became professional, it took a disciplined approach with education rather than training.

Customer needs have increased, projects have become advanced, and the attraction of the people has increased toward s/w development.

The general stages of engineering s/w include feasibility study and preliminary investigation, requirements analysis and specification, design, coding, testing, deployment, operation, and maintenance.

Each stage employs some methodology and tool for better quality products and effective s/w development.

S/W Crisis:

Software crisis, the symptoms of the problem of engineering the s/w, began to enforce the practitioners to look into more disciplined s/w engineering approaches for s/w development.

The modern software crisis has some notable symptoms, which are complexity, h/w versus s/w cost, lateness, costliness, poor quality, unmanageable nature, immaturity, lack of planning and management practices etc.,

An important cause of s/w crisis is the complexity of s/w development process.

As programs became more complex, earlier methodologies of s/w development were no longer useful for larger and complex projects.

Designing s/w is a labour-intensive task; hence, with a growing economy, more wages are to be paid. Development, maintenance, change and reengineering processes require logical thinking and labour work. Ultimately the cost of s/w is higher than the h/w cost.

The time required to develop s/w and its cost began to exceed all estimates. It is common for the system to cost more than what had been estimated at the time of project planning (lateness, costness).

S/W must be of high quality with respect to product operation, product transition and product migration.

The programmers productivity must be increased.(quality)

In early days of development, change in s/w is a major problem for s/w practitioners and it makes around 40% of the total development cost.

After changing a code, the entire s/w or subsystem needs to be tested for reliable functioning of the system. Therefore, some systematic engineering approaches are needed for maintenance and changes.

In the modern heterogeneous development of s/w, different people are involved in the project. The project must be planned with the expected requirement of resources (h/w, s/w, people etc.), cost, time and effort.

S/W crisis has inspired people to improve the processes. IT practitioners always try to follow some formal process that can help them to produce quality products, reduce cost, and improve team communication & morale.

There are several such problems in the s/w industry. S/w products become costly, are delivered late, are unmanaged, have poor quality, decrease the productivity of programmers, increase the maintenance cost and rework, and lack of nature s/w processes in a complex project.

The solution to these s/w crises is to introduce systematic s/w engineering practices for systematic s/w development, maintenance, operation, retirement, planning and management of s/w.

What is S/W Engineering:

Definition: S/W Engineering is an engineering, technological, and managerial discipline that provides a systematic approach to the development, operation, and maintenance of s/w.

Engineering provides a step-by-step procedure for s/w engineering i.e., project planning, problem analysis, architecture and design, programming, testing & integration, deployment and maintenance and project management.

These activities are performed with the help of technological tools that ease the execution of above activities.

Management skills are necessary in a project manager to communicate and coordinate the information and management of these activities.

IEEE Definition:

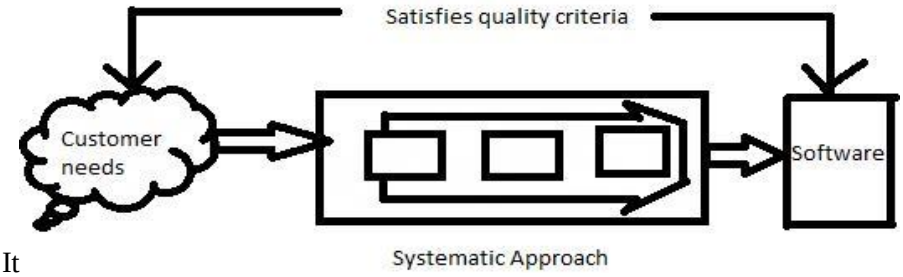
The systematic approach to the development, operation, maintenance and retirement of s/w.

"The main goal of s/w engineering is to understand customer needs and develop s/w with improved quality, on time, and within budget.

Along with the development of s/w intensive system, industries are also focusing on increasing productivity, improving the process, reducing rework, decreasing development cycle time, and developing an adaptable system.

S/W architecture is selected based upon the stakeholder's need, operational environment of the developed S/W, and technology awareness of the development team.

S/W Engineering View



It

Evolution of S/W Engineering Methodologies:

It is a set of procedures followed from the beginning to the completion of the development process. S/W Engineering Methodologies have evolved with increasing complexities in programming and advancements in programming technologies. Those are:

1. Exploratory Methodology:

   This style of methodology is applied whenever the requirements are initially unclear. Involves experimentation and exploring the programs through step-by-step programming

   The process of each step depends on the previous ones.

   Errors are detected only during the final product testing.

2. Structure-Oriented Methodology:

   Focuses on procedural approach, which concentrates on developing functions or procedures, i.e., the programmer writes out the instructions that are followed by computer from start to finish.

   - Three basic elements are Sequence, selection and iteration.
   - It uses variety of notations such as Data Flow Diagrams(DFD), CFG,ER etc.,

   This approach is preferred in scripts and embedded systems with small memory requirements with high speed.

3. Data-Structure – Oriented Methodology:
   - Concentrates more on designing data structures rather than on procedures and control.
   - Helps to determine the program structure because data play an important role in the execution of a program.

- Jackson Structured Design (JSD) methodology developed by Michael Jackson in 1970 is a famous data-structure oriented methodology which explains functionality in the real world.

4. Object-Oriented Methodology:
   - Emphasizes the use of data rather than functions. Real world entities are treated as objects.
   - Data & Procedures are built around these objects.

   Object involves properties & methods where

   i) Properties are nothing but characteristics which define how an object behaves.
   ii) Methods are blocks of instructions that can be executed by an object itself, and each object has its own set of methods.

5. Component-Based Methodology:

   When the programming is of small context, then flow of control and data structures are the concerns which focuses on its algorithm & data structures.

   When it is of large programming context, then the major concern is about components and their integration.

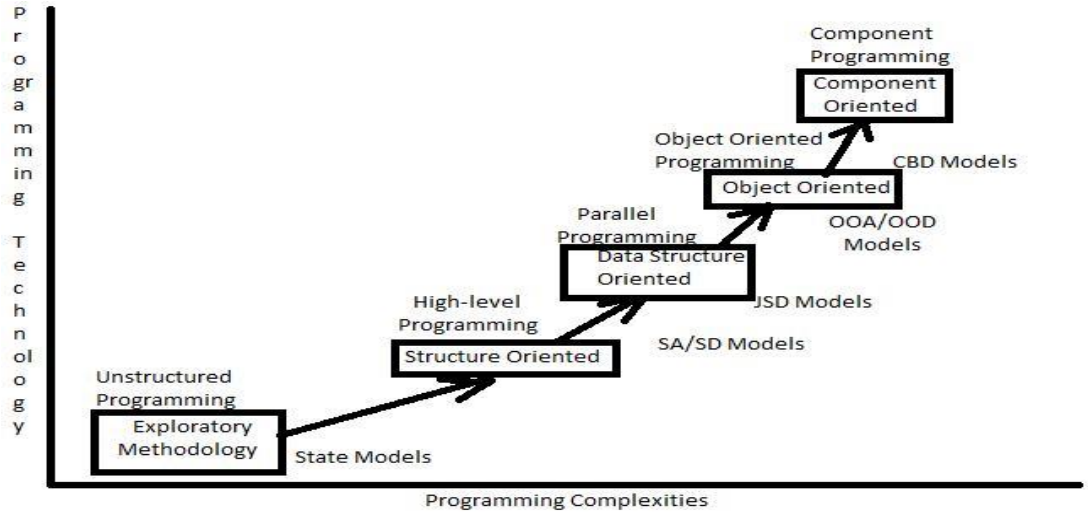
   It became a significant methodology for communication among different stakeholders and for large-scale reuse.

   CBD was introduced in 21st century which is a system analysis and design methodology that has evolved from the object-oriented Methodology.

   It is largely based on its focus of reuse.

   It emphasizes reuse in which components represent coherent parts of a system that can be independently stored and assembled into S/W systems.

Pictorial Representation of s/w eng. Methodologies evolution:



S/W Engineering Challenges:

S/W Engineering is facing a number of s/w problems since s/w crisis. The primary focus of s/w companies is to produce quality s/w within budget and small cycle time. These s/w challenges reflect the development and maintenance of s/w is an important issue or practitioners.

1. Problem Understanding:

   Customers are from different backgrounds and they do not have a clear understanding of their problems and requirements. Also, the customers don't have technical knowledge, especially those who are living in remote areas.

   Similarly s/w engineers do not have the knowledge of all application domains and detailed requirements of the problems and the expectations of the customer. The lack of communication among s/w engineers and customers causes problems for the s/w engineers in clearly understanding the customer needs. There are many people involved in an organization. Therefore, it becomes difficult to find requirements from different perspectives and customers. Sometimes the customers do not have sufficient time to explain their problems to the development organization.

2. Quality and Productivity:

   S/W Engineering practices mainly emphasize providing quality products in a small cycle time.

   Quality products provide customer satisfaction.

   A good quality product implements features that are required by the customer that have certain quality attributes such as reliability, usability, efficiency, maintainability, portability, and functionality.

   Production of S/W is measured in terms of KLOC person month (PM).

   Higher productivity means that cycle time can be reduced with the low cost of the product.

   The productivity and quality of S/W depend on several factors, such as programmer's ability, type of technology, level of experience, nature of projects and their complexity, available time, required resources etc.,

3. Cycle Time and Cost:

   The customer always foresees faster and cheaper production of S/W products. Therefore, s/w companies put efforts to reduce the cycle time of product delivery and minimize the product cost.

   Delivery before the calendar time sometimes compromises the product quality due to competitive reasons.

   The cost of the s/w product is generally the cost of H/W, S/W and manpower resources.

   It is calculated based on the no. Of persons engaged in a project and for how much time.

   Cost also depends on the complexity levels of the project.

4. Reliability:

   It is the most important quality attribute.

   It is the successful operation of s/w within the specified environment and duration under certain constraints.

   Verification and Validation techniques are used to ensure the reliability ratio in the product.

   Defect detection and prevention is the prerequisite to high reliability in the product where several automated testing tools are available for its removal.

   S/W becomes unreliable due to logical errors present in the programs of the S/W. Project complexity is the major cause of s/w unreliability.

5. Change and Maintenance:

   Change and maintenance in s/w come when the s/w is delivered and deployed at the customer site.

   They occur if there is any change in the business operation, errors in the s/w, or addition of new features.

   Change in one part of the s/w requires change in other parts also. After any change and maintenance operation, s/w is rigorously tested to keep it reliable.

   As change and maintenance in S/W are flexible but very expensive, sometimes the maintenance and rework cost becomes more than the development cost.

   The challenge is to accommodate changes under controlled cost and reliability as due to repeated maintenance and change, s/w deteriorates its operational life and quality.

   Thus, to accommodate increasing requirements and streamline the modern technology, s/w is needed to be reengineered onto a modern platform.

6. Usability and Reusability:

   Usability means the ease of use of a product in terms of efficiency, effectiveness and customer satisfaction.

   S/W Engineering has always concentrated on providing a usable product by incorporating customer suggestions and technological issues.

   There are many products which came into existence but became unsuccessful due to certain s/w design and usability issues.

   Reuse of existing s/w components and their development has become institutional business in the modern s/w business scenario.

   The analysis of domain knowledge, development of reusable library, and integration of reusable components in s/w development are some important issues in reuse-based development.

   Reusability increases reliability because reusable components are well tested before integrating them into s/w development.

7. Repeatability and Process Maturity:

   S/W development processes are procedural rules to adhere to when developing s/w.

   A S/W engineering process can be repeated in similar projects, which improves productivity and quality.

   Repeatability can help to plan project schedule, fix deadlines for product delivery, manage configuration, and identify locations of bug occurrences.

   Repeatability promotes process maturity.

   A Mature S/W process produces quality products and improves S/W productivity.

There are several standards, such as CMM, ISO and six sigma, which emphasize process maturity and its guidelines.

8. Estimation and Planning:

   Present estimation methods, such as Lines of Code (LOC), Function Point (FP), and Object Point (OP), are sometimes unable to accurately estimate project efforts.

   Most of the projects fail due to underestimation of budget and time to complete the project.

   The effectiveness of the project plan depends on the accuracy of the estimation and understanding of the problem.

   The use of an effective project planning and estimation technique is an important challenge for the practitioners.

   The rest of the project's activities depend on estimation and planning.

Software Process:

Introduced in 1950's – 1960's

The developers were using ad-hoc processes for s/w development due to alternative technologies and lack of enterprise and resources.

So, for the systematic approach, we introduce a S/W Process.

i) **A) Definition:** It is a set of ordered activities carried out to produce a S/W product.

Each activity has well-defined objective, task and outcome.

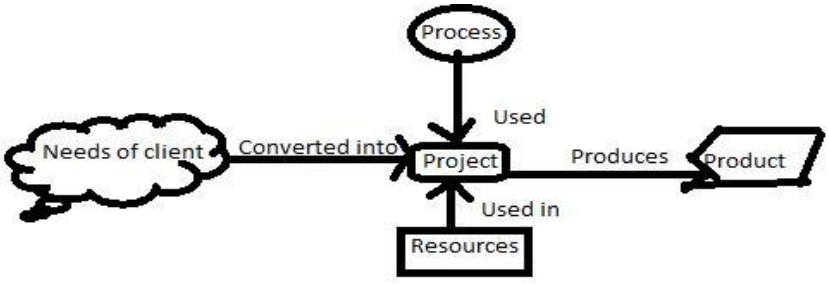
A activity is a specified task performed to achieve the process objectives. The outcomes of all activities are compiled and integrated together to design the s/w. The development of S/W is done with the help of some s/w process methodologies.

Each activity of a S/W process involves tools and technologies (CASE tools, compiler etc), procedures (algorithms, installation procedure etc.) and artifacts (the intermediate of final outcomes).

B) Software Project: It is an entity. With defined start and, in which a S/W process is being used. Type of the process and its execution depend on the nature of project.

C) Software Product: It is the outcome of the s/w project produced through processes. A project can have more than one product called work products. A work product is the intermediate outcome of the processes. But the final work product is referred to as a product or S/W.

Diagrammatical Representation of the Relationship b/w Process, Project and Product:



ii) S/W Process Model:

It is a generic representation of a s/w process instantiated for each specific project. A process model is a set of activities that have to be accomplished to achieve the process objectives.

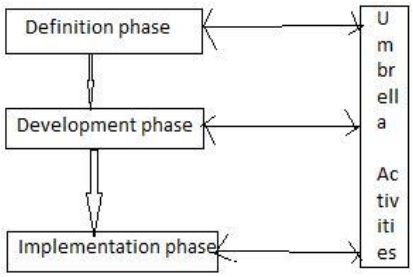
It reduces chaos of s/w development even though it is difficult to execute in the real world as it is the idealization of the S/W process model.

A particular process model can be useful in one context and may be less useful in another context. Thus, different process models are generally needed o develop and understand the various aspects of projects.

These models may be related to development, management, improvement, and maintenance.

Examples are: Data flow model, life cycle model, quality model etc.,

Generic view of a S/W Process model



The generic process model has three phases that are coordinated and supported by umbrella activities. The phases in a process model are:

1. Definition Phase:

   It concentrates on understanding the problem and planning for the process model. The activities may include problem formulation, Problem analysis, system engineering, and project planning for the process.

2. Development Phase:

   i) It focuses on determining the solution of the problem with the help of the umbrella activities.

   ii) The main activities of this phase are designing the architecture and algorithms of the system, writing codes, and testing the S/W.

3. Implementation Phase:

Deployment, change management, defect removal, and maintenance activities are performed in this phase.

Reengineering may take over due to the changes in the technology and business.

The umbrella activities are responsible for ensuring the proper execution of definition, development, and implementation phases.

The umbrella activities are project management, quality assurance, configuration management, risk management, work products preparation & deployment, and process improvement.

Various reviews and rigorous testing must be carried out through quality assurance procedures.

iii) Elements of S/W Process:

It comprises of various essential elements. These elements are used together to produce a product. Those are:

1. Artifacts: These are tangible work products produced during the development of s/w. These are specified in advance in the development process so that the activity could be performed accordingly and it can be used as raw artifacts to generate other artifacts.
   Ex: S/W Architecture, Project plan etc.,
2. Activity: Specifies the tasks to be carried out implicitly or explicitly. Each activity receives some i/p, executes the task on the laid constraints, and produces certain work products that can be used as input for some other activity. Each activity uses some procedures, rules, policies and guidelines to produce the required artifacts.
   Ex: Analysis, Design, Tracking and Monitoring etc.,
3. Constraint: Refers to the criteria or condition that must be met or possessed by a s/w product.
   Ex: A M/C allows five users to login at a time, permits seven transaction per nanosecond etc.
4. People: People or persons or stakeholders who are directly or indirectly involved in the process. Stakeholders play important roles in achieving project goals.
   Ex: S/W tester, quality checker etc.,
5. Tools and Technology: It provides technical support to the methods or techniques to be used for performing the activities.
   Ex: FORTRAN – Scientific problems, CASE tools – S/W development, BASIC – business
6. Methods or Techniques: It specifies the way to perform an activity using tools and technology to accomplish the activity. It provides detailed mechanism to carry out an activity.
   Ex: Object Oriented Analysis (OOA), binary search etc.,
7. Relationship: It specifies the link among various activities or entities. It assists in the execution sequence of activities where the outcome of an activity can be used as an i/p to the subsequent activity.
   Ex: Maintenance followed by implementation, debugging is required after error detection etc.,
8. Organizational structure: It specifies the team of people that should be coordinated and managed during s/w development. All organizations have a structure with defined roles and responsibilities of individuals. Organizational structure reflects the success of s/w projects and processes.
   Ex: the project leader monitors the work flow of various activities which are assigned to the s/w engineers.

iv) Characteristics of S/W Process:

It is very difficult to identify a s/w process for the process of development without understanding the attributes and scope of the specified s/w process. For example, chat server, real time operating system, large payroll system etc. Need a different kind of process. However some characteristics will be common, those are:

1. Understandability: The process specification should be easy to understand, easy to learn and easy to apply. Thus, the process should be specified with its adaptation procedure, execution mechanism, supporting tools and technology.
2. Effectiveness: The produced product should adhere to the schedule and quality constraints. Effectiveness of a process depends on certain performance indicators, such as programmer's skills, fund availability, quality of work products, etc.
3. Predictability: It is about forecasting the outcomes before the completion of a process. It is the basis through which the cost, quality, and resource requirements are specified in a project. The quality of a product and its deliverables can be expected according to the success rate of the process. Sometimes a predictable process is referred to as being under statistical control process. Under statistical control, a process can produce outcomes as per the expected input values.
4. Maintainability: It is the flexibility to maintain s/w through change requirements, defect detection and correction, adopting it in new operating environments, which is a life-long process and sometimes its cost exceeds the actual s/w development cost. It is one of the primary objectives of a process to reduce a maintenance task in s/w which in-turn reduces project cost.
5. Reliability: It refers to the capability of performing intended tasks. Rigorous testing procedures are carried out by applying a process in any production process. Unreliability of a process causes failures and unreliable processes waste time and money.
6. Changeability: It is the acceptability of changes done in s/w. It is classified into robustness, modifiability, and scalability.
    i) Robustness: Process that does not change the product quality due to its internal and external changes.
    ii) Scalability: Ability to change the attributes so that the process can be used in smaller to larger s/w development.
    iii) Modifiability: Ability of adoptability of change occurrence.
7. Improvement: It is a process which helps to enhance quality of the delivered products for providing more satisfactory services to the users. There are various process improvement standards, such as quality improvement Paradigm (QIP), Capability Maturity Model Integration (CMMI) etc.
8. Monitoring and Tracking: It is a process in a project can help to determine predictability and productivity. Helps to monitor and track the progress of the project based upon past experiences of the process. The feedback from the earlier projects and products is considered during the assessment and improvement in the process.
9. Rapidity: It is the speed of a process to produce the products under specifications for its timely completion. Understandability and tracking of the process can accelerate the production process.
10. Repeatability: It measures the consistency of a process so that it can be used in various similar projects. A process is said to be repeatable if it is able to produce an artefact no. Of times without the loss of quality attributes. There may be variations in the operation, cost, and time but the quality of artifacts will be the same.

Process Classification:

S/W processes may be classified as:

1. **Product Development Process:**
   Product development processes focus mainly on producing s/w products. They are the core process in any s/w project. These processes involve various techniques, tools, and technologies for developing s/w.
   Processes include various activities like conceptualization, designing, coding, testing, and implementation of a new or existing system.
   There are certain work products of these activities, such as SRS, design models, source codes, test reports, and documentation. Thus, the people with different skills are involved in product development processes.
   The most widely used s/w development process models are the waterfall model, prototyping model, spiral model, agile model, RUP, and so on.
   The applicability of these models depends upon the application, constraints, and the nature of the project.
   Customer feedback, reusability, coordination, communication, and documentation are some factors that help to decide the application of development process models.
2. **Project Management Process:**
   It concentrates on planning and managing projects in order to achieve the project objectives. The goal of these processes is to carry out the development activities within time, budget, and resources.
   Various project management processes are Initiating, planning, coordinating, controlling, executing, and terminating are the main activities of a general project management process which have been taken place under project manager.
   The project manager designs teams, allocate the tasks, and monitors the progress of the project team members so that the project could be completed on time and within budget.
3. **Process improvement process:**
   These processes are involved in improving the process itself. The ultimate goal of improvement in a process is to enable the organization to produce more quality products.
   It may happen that a process which is successful in some projects may be unsuccessful in other projects. Process improvement is an incremental improvement of process, which is used for s/w development.
   There exist various process improvement process models, such as CMMI, QIP, CQI-Continuous Quality Improvement, TQM- Total Quality Management, Six Sigma and so on which will provide improvement guidelines and standards for improving s/w processes.
4. **Configuration Management Process:**
   Changes may occur in projects, processes and products as these entities are evolutionary in nature. Changes may arise due to either change in the customer requirements or discrepancies in the work products or procedures from the developer's side.
   Identifying, evaluating, and finally implementing changes is the main function of s/w configuration management (SCM) process. During this process, integrity of the product, process, and project is maintained throughout its life cycle.

Configuration Management includes various activities for performing changes, such as identification of configuration items, devising mechanisms for performing changes, controlling changes, and tracking the status to those changes.

Successful configuration management can reduce the overall development cost and enhance product quality and integrity.

5. **Quality Management Process:**

The ultimate goal of all the above processes is to develop a quality s/w product.

A quality management process provides metrics, feedback, and guidelines for the assurance of product quality.

The main activities of s/w quality groups are verification & validation, acceptance testing, measurement and metrics, process consulting, and so on.

ISO 9000 is a framework that provides certain guidelines for quality system.

Phased Development Life Cycle:

Product development is a complex and long-running process whose aim is to produce quality S/W products.

Each activity in the process is also referred to as a **phase**. Each phase in the process again acts as a process for performing the specified activity. General activities include feasibility study, analysis, design, coding, testing, implementation, and maintenance.

The above activities are collectively called as Software Development Life Cycle(SDLC) or software life cycle and each of these activities is called Life Cycle Phase.

Each phase has some defined initiating and completion criterion which produces some kind of intermediate outcomes (work products) for product development.

Examples of these models are waterfall, prototyping, spiral, incremental, agile process, RUP process model, and so on.

Each model begins with customer needs, performs life cycle activities, and ends with some work products.

Each process model has a workflow in which these activities are interrelated to one another.

These activities are undertaken in sequence whereas in other models, activities may be in repeated in a closed loop.

In large and complex projects, activities may be carried out implicitly.

Phased Life Cycle Activities:

The general development process activities which are covered in software development life cycle models are feasibility study, requirements analysis, design, code, testing, deployment, operation, and maintenance.

The developed s/w is used in the environment and faults may be observed during its operation.

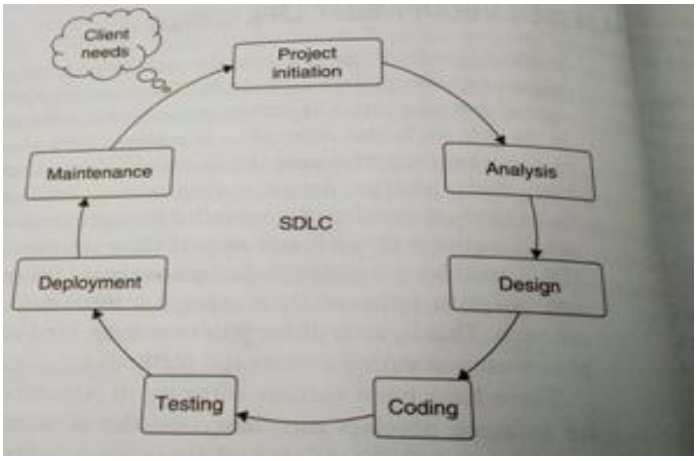
It is observed that the maintenance cost sometimes exceeds with the development cost due to repeated maintenance, increased user needs, and rapid growth of technology.

1.Project Inititation:

It involves preliminary investigation, feasibility study, and a project plan. Preliminary Investigation (PI) is the initial step that gives a clear picture of what actually the physical

system is. PI goes through problem identification, background of the physical system, and the system proposal for a candidate system. On the basis of this feasibility study is performed.

SDLC Activities



The purpose of feasibility study is to determine whether the implementation of the proposed system will support the mission and objectives of the organization. It ensures that the candidate system is able to satisfy the user needs; promotes operational, effective use of resources; and is cost effective.

It is classified as Technical, economical, operational feasibility. Economic feasibility involves Cost/Benefit Analysis. A feasibility report is prepared and submitted to top-level management. A positive report leads to project initiation. A detailed project plan is actually prepared after knowing the requirements.

2. Requirements Analysis:

Process of collecting factual data, understanding the process involved, defining the problem, and providing a document for further software development.

It consists of three main activities:

1) Requirements Elicitation: About understanding the problem
2) Requirements Specification: Requirements Specification Document has been prepared in this described which an SRS is.
3) Requirements Verification and Validation: To check whether the correct requirements are stated (validated) and that these requirements are stated correctly (verification).

3. Software Design: The goal of the design phase is to transform the collected requirements into a structure that is suitable for implementation in programming languages. The S/W designers begin with making architectures, outlining the hierarchical structure, and writing algorithms for each component in the system. The two aspects are physical design and logical design

4. Coding: It is concerned with the development of source code that will implement the design. Good coding efforts can reduce testing and maintenance tasks. Programs must be modular so that they can help in rapid development, maintenance, and enhancements of the system. Rules must be followed for the declaration of data structures, variables, header files, function call and so on.

5. Testing: Testing is performed to remove the defects in the developed system. Test plan of the system is developed and run on the specified test data. Testing covers various errors at the requirements, design, and coding phases. Testing is performed at different levels: unit testing, integration testing, system testing and acceptance testing.

Unit testing is carried out for individual modules at the code level.

After testing each module, interfaces among various models are checked with integration testing.

System test ensures that the system satisfies the requirements specified by the customer.

Acceptance test is done for customer satisfaction.

6. Deployment: After acceptance by the customer during the testing phase, deployment of the software begins. The purpose of the software deployment is to make the software available for operational use. The release of the software starts. Required resources are procured to operate at the customer site and important programs files are loaded onto user's computer. After installation of all modules of the system, training of the user starts.

Documentation of the system is also an important activity in software development. Documentation is in the form of a user manual or system operation process which ensures the continuity of the system. User documentation is the description of the system from the user's point of view, detailing how to use or operate the system.

7. Maintenance: It comes after the software product is released and put into operation through the deployment process. Is performed to adapt to changes in a new environment, correct bugs, and enhance the performance by adding new features. Its activities can be classified as adaptive, perfective, corrective, and preventive.

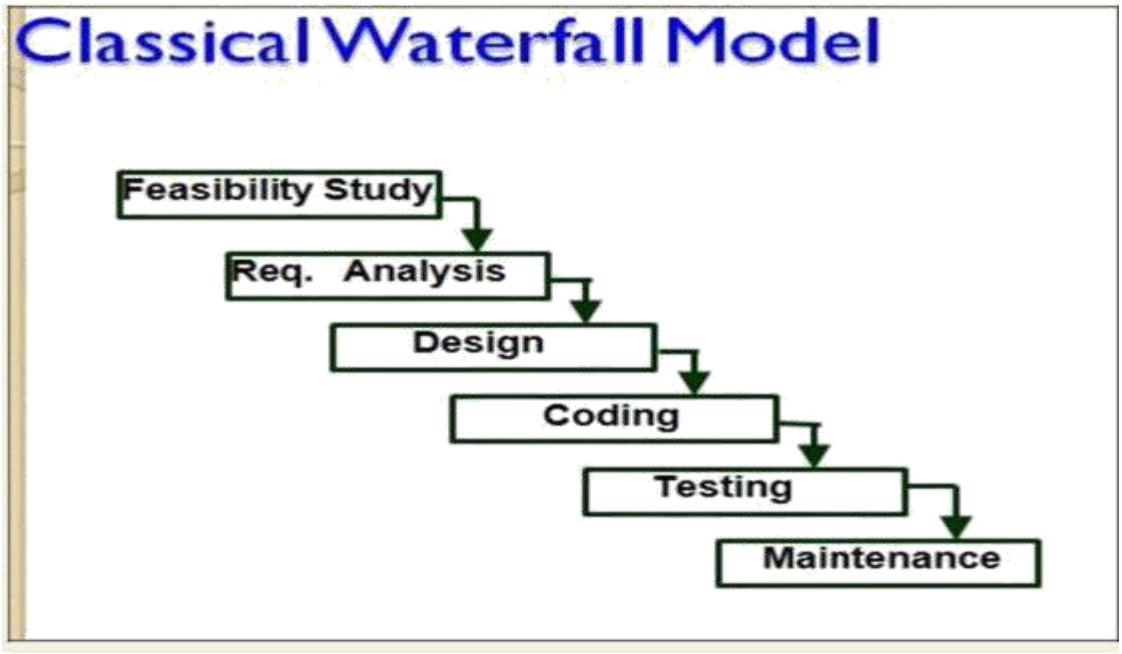
Software Development Process Models:

Various S/w process models have been proposed due to the varying nature of software applications. The process models which we will discuss are:

1. Classical Waterfall Model
2. Iterative waterfall model
3. Prototyping model
4. Incremental model
5. Spiral model
6. Agile Process model
7. RUP process model

1. Classical Waterfall Model:

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of **software development model** is basically used for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model **software testing** starts only after the development is complete. In **waterfall model phases** do not overlap.



Advantages of waterfall model:

This model is simple and easy to understand and use.

It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.

In this model phases are processed and completed one at a time. Phases do not overlap.

Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the **testing** stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects. Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed. Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely the project is short.

Very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demoted to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.

2. Iterative Waterfall Model

To overcome the major shortcomings of the classical waterfall model, we come up with the iterative waterfall model.

Here, we provide feedback paths for error correction as & when detected later in a phase. Though errors are inevitable, but it is desirable to detect them in the same phase in which they occur. If so, this can reduce the effort to correct the bug.

The advantage of this model is that there is a working model of the system at a very early stage of development which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

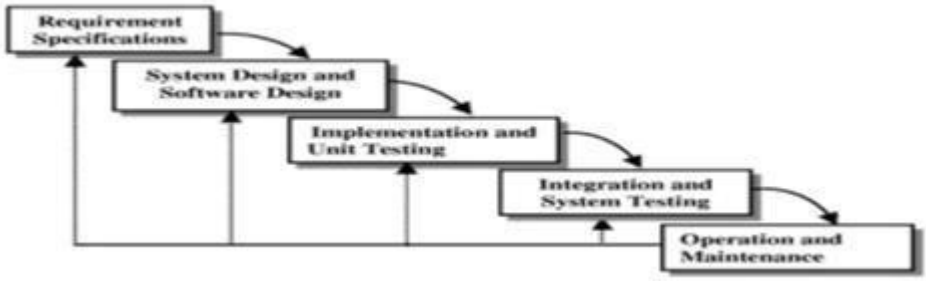


Fig : Iterative Waterfall Model

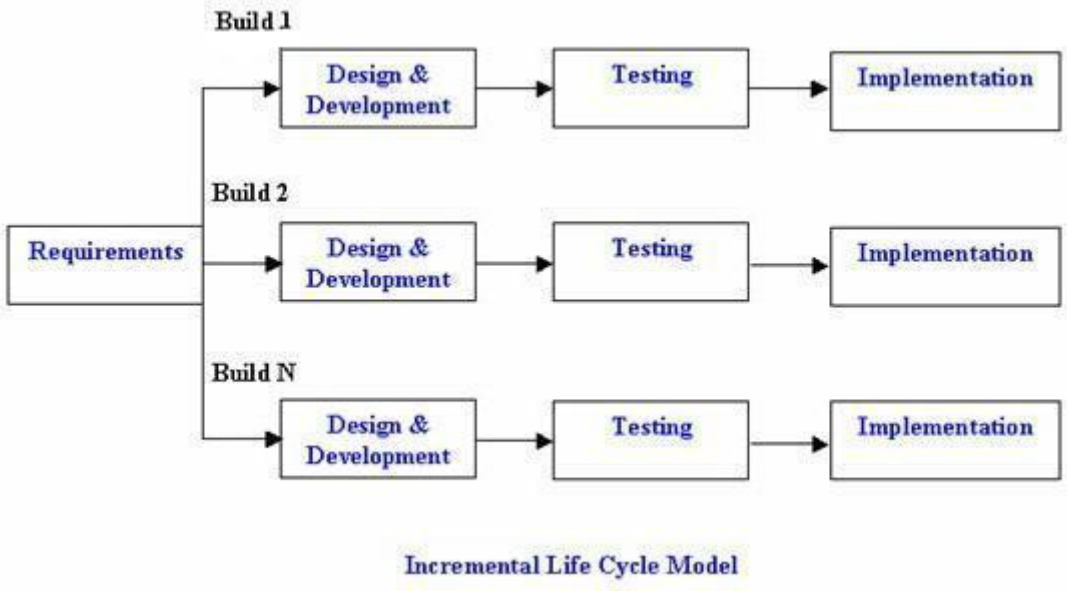
3. Incremental Model:

In this model, each module passes through the requirements, design, implementation and **testing** phases. A working version of software is produced during the first module, so you have working software early on during the **software life cycle**. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

For example:

In the diagram above when we work **incrementally** we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it's complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoted to the customers. Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

Diagram of Incremental 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'd 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:

- 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.
- A new technology is being used
- Resources with needed skill set are not available. There are some high risk features and goals.

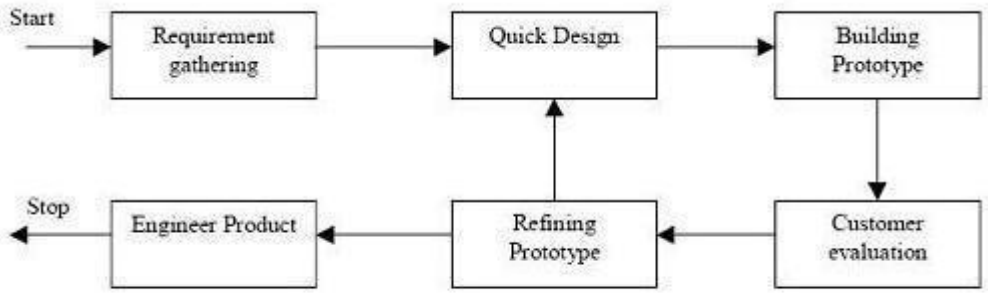
4. <u>Prototype Model:</u>

The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a **software development model**. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the

desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

The prototypes are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:



Prototyping Model

Advantages of Prototype model:

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

Confusing or difficult functions can be identified in each and every iteration,

Requirements validation, Quick implementation of incomplete but functional and application model.

Disadvantages of Prototype model:

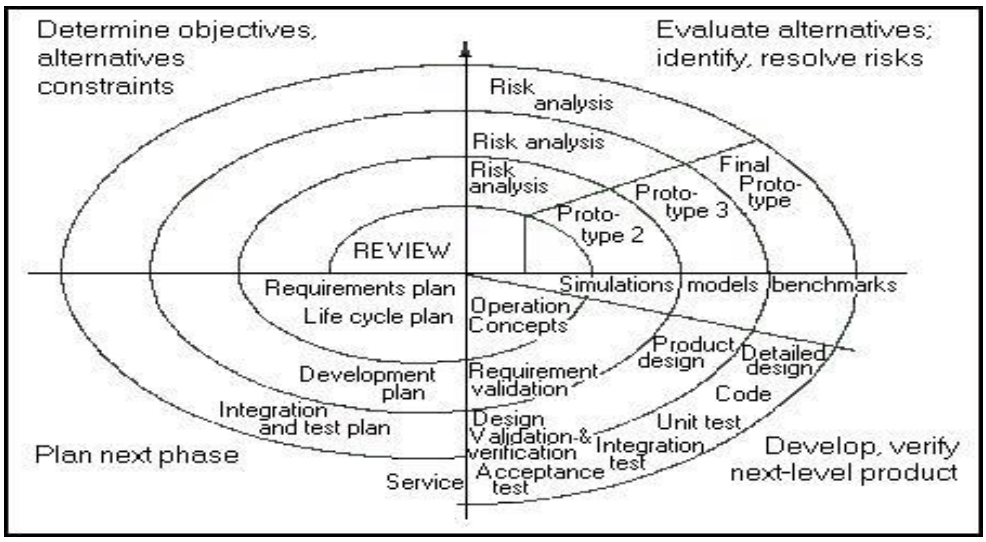
- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may result in not using of that application. Incomplete or inadequate problem analysis.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

5. Spiral Model:

The Spiral model of software development is shown in the figure given below. The diagrammatic representation of this model appears like a spiral with many loops. The exact number of loops in the spiral is not fixed. Each loop of the spiral represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study, the next loop with requirements specification, the next one with design, and so on. Each phase in this model is split into four sectors (or quadrants) as shown in the below figure. The following activities are carried out during each phase of a spiral model.



First quadrant (Objective Setting)

- During the first quadrant, it is needed to identify the objectives of the phase.
- Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

- A detailed analysis is carried out for each identified project risk.
- Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

Third Quadrant (Development and Validation)

- Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

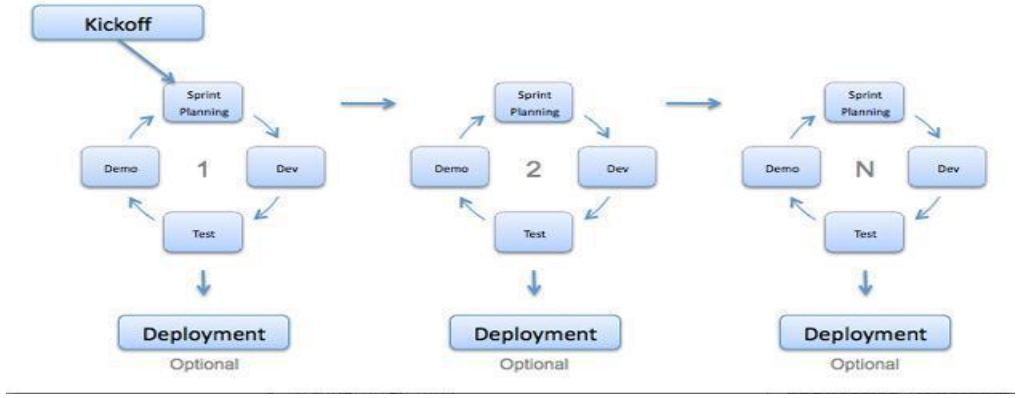
- Review the results achieved so far with the customer and plan the next iteration around the spiral.
- Progressively more complete version of the software gets built with each iteration around the spiral.

Circumstances to use spiral model

The spiral model is called a Meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models – this is probably a factor deterring its use in ordinary projects.

6. Agile Process Model:

Agile development model is also a type of **Incremental model**. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly **tested** to ensure **software quality** is maintained. It is used for time critical applications. Extreme Programming (XP) and Scrum are currently the most well known agile **development life cycle model**s.



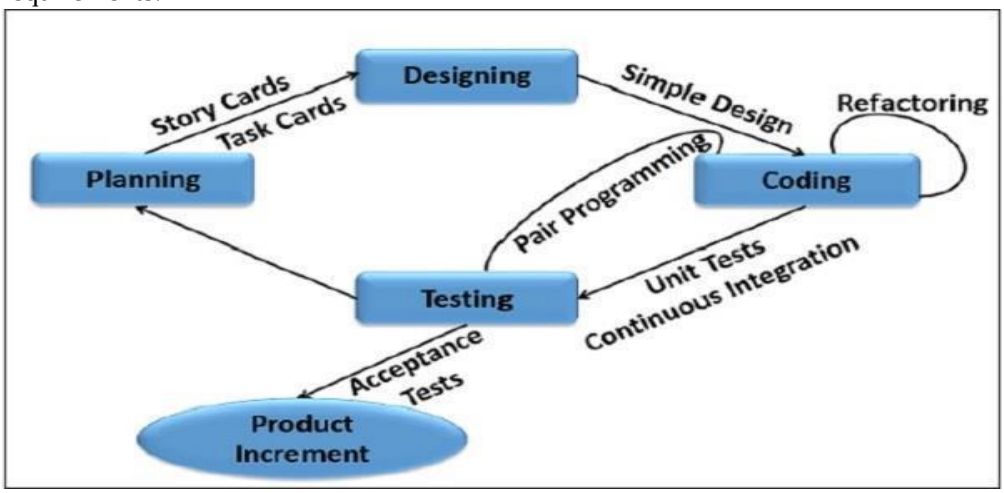
Extreme Programming:

XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software. eXtreme Programming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements. Extreme Programming is one of the Agile software development methodologies. It provides values and principles to guide the team behavior. The team is expected to self-organize. Extreme Programming provides specific core practices where- Each practice is simple and self-complete. Combination of practices produces more complex and emergent behavior.

Extreme Programming in a Nutshell Extreme Programming involves-

- Writing unit tests before programming and keeping all of the tests running at all times. The unit tests are automated and eliminate defects early, thus reducing the costs.
- Starting with a simple design just enough to code the features at hand and redesigning when required.
- Programming in pairs (called pair programming), with two programmers at one screen, taking turns to use the keyboard. While one of them is at the keyboard, the other constantly reviews and provides inputs.
- Integrating and testing the whole system several times a day.

- Putting a minimal working system into the production quickly and upgrading it whenever required.
- Keeping the customer involved all the time and obtaining constant feedback. Iterating facilitates the accommodating changes as the software evolves with the changing requirements.



Advantages:

- **Slipped schedules:** Short and achievable development cycles ensure timely deliveries.
- **Cancelled projects:** Focus on continuous customer involvement ensures transparency with the customer and immediate resolution of any issues.
- **Costs incurred in changes:** Extensive and ongoing testing makes sure the changes do not break the existing functionality. A running working system always ensures sufficient time for accommodating changes such that the current operations are not affected.
- **Production and post-delivery defects:** Emphasis is on the unit tests to detect and fix the defects early.
- **Misunderstanding the business and/or domain:** Making the customer a part of the team ensures constant communication and clarifications.
- **Business changes:** Changes are considered to be inevitable and are accommodated at any point of time.
- **Staff turnover:** Intensive team collaboration ensures enthusiasm and good will. Cohesion of multi-disciplines fosters the team spirit.

The fundamental principles of Extreme Programming are-

- Rapid feedback
- Assume simplicity
- Incremental change
- Embracing change
- Quality work

Scrum Programming:

Scrum is an agile way to manage a project, usually software development. Agile software development with Scrum is often perceived as a methodology; but rather than viewing Scrum as methodology, think of it as a framework for managing a process.

Scrum Overview - Introduction to Scrum Terms

An introduction to Scrum would not be complete without knowing the Scrum terms you'll be using. This section in the Scrum overview will discuss common concepts in Scrum.

Scrum team: A typical scrum team has between five and nine people, but Scrum projects can easily scale into the hundreds. However, Scrum can easily be used by one-person teams and often is. This team does not include any of the traditional software engineering roles such as programmer, designer, tester or architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Scrum teams develop a deep form of camaraderie and a feeling that "we're all in this together."

Product owner: The product owner is the project's key stakeholder and represents users, customers and others in the process. The product owner is often someone from product management or marketing, a key stakeholder or a key user.

Scrum Master: The Scrum Master is responsible for making sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside, and so on.

Product backlog: The product backlog is a prioritized features list containing every desired feature or change to the product. Note: The term "backlog" can get confusing because it's used for two different things. To clarify, the product backlog is a list of desired features for the product. The sprint backlog is a list of tasks to be completed in a sprint.

Sprint planning meeting: At the start of each sprint, a sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team. The Scrum team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint.

Daily Scrum: Each day during the sprint, a brief meeting called the daily scrum is conducted. This meeting helps set the context for each day's work and helps the team stay on track. All team members are required to attend the daily scrum.

Sprint review meeting: At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which, the team shows what they accomplished during the sprint. Typically, this takes the form of a demonstration of the new features, but in an informal way; for example, PowerPoint slides are not allowed. The meeting must not become a task in itself nor a distraction from the process.

Sprint retrospective: Also at the end of each sprint, the team conducts a sprint retrospective, which is a meeting during which the team (including its Scrum Master and product owner) reflect on how well Scrum is working for them and what changes they may wish to make for it to work even better.

Scrum is an iterative framework to help teams manage and progress through a complex project. It is most commonly used in Software Development by teams that implement the Agile Software Development methodology. However it is not limited to those groups. Even if your team does not implement Agile Software Development, you can still benefit from holding regular scrums with your teams.

Scrum participants fall into the same two categories. They are either Pigs or they are chickens. Participants at scrum are either fully committed to the project or simply participants. Let's look at who these various roles really are.

Pig Roles

- **Actual Team Members:** These would be the developers, artists or product managers that comprise the core of the team. These are the people who are actually doing the daily work to bring the project to fruition. These members are fully committed to the project.
- **Scrum Master:** The scrum master might be one of the team members — or might not be. It is important to call this person out separately here though because the Scrum master has the primary role of ensuring that the scrum moves forward without problems and is effective for the team.
- **Project Owner:** This may be a Product Manager who is also comprised of the team or it may not. Again it is important to call this persons role out here as this person represents the voice of the end customer. This person needs to ensure that the product achieves it's product goals and provides the necessary end product to the customers.

Chicken Roles

- **Managers:** At first glance you might think that managers are pigs — naturally. However in the scrum context managers are generally more concerned about the people involved in a project and their respective health. They are not as focused on the product and it's particular customer oriented goals. For this reason they are considered a chicken in the scrum context.
- **Stakeholders:** Stakeholders are individuals who will benefit or have a vested interest in the project, however do not necessarily have authority to dictate direction or to be held accountable for the product. They can be consulted for opinions and insight however the product owner needs to maintain final rights for the decision making process.

Why are the roles important

The chicken and pig roles are vital to scrum because it dictates who in the scrum should be an active participant Chickens should not be active participants in a scrum meeting. They may attend, however they should be there as guests only and not required to share their current statuses. Pigs on the other hand need to share their current progress and share any blockers that they are encountering.

The reason that Chickens should not be active participants is that they too easily will take over the direction of the scrum and lead it away from the goals of the entire team. It is the scrum masters job to ensure that the scrum stays on target and covers the topics that need to be covered. if someone goes off topic (chicken or pig) it is the scrum masters job to bring the group back to the topic at hand.

Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months). Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers. Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use Agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the **waterfall model** in agile model very limited **planning** is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

Rational Unified Process Model:

Stands for "Rational Unified Process." RUP is a software development process from Rational, a division of IBM. It divides the development process into four distinct phases that each involve 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.
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.
3. **Construction** - The project is developed and completed. The software is designed, written, and tested.
4. **Transition** - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

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 resources from being wasted and reduces unexpected development costs.

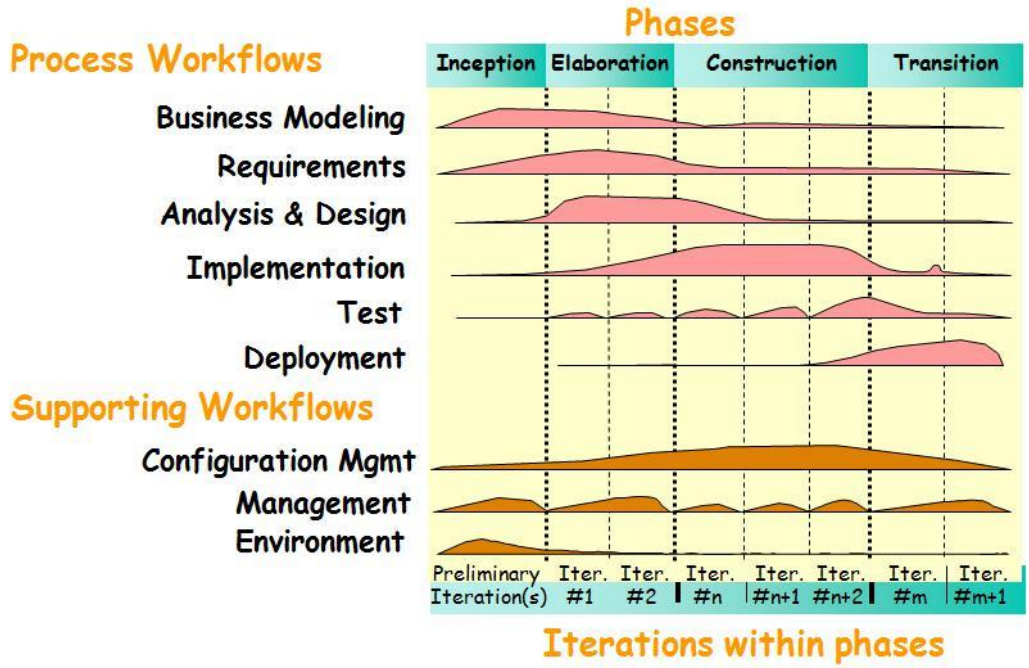
Advantages of RUP Software Development

1. This is a complete methodology in itself with an emphasis on accurate documentation
2. It is proactively able to resolve the project risks associated with the client's evolving requirements requiring careful change request management
3. Less time is required for integration as the process of integration goes on throughout the software development life cycle.
4. The development time required is less due to reuse of components.
5. There is online training and tutorial available for this process.

Disadvantages of RUP Software Development

1. The team members need to be expert in their field to develop software under this methodology.
2. The development process is too complex and disorganized.
3. On cutting edge projects which utilise new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfil.
4. 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.

As seen, RUP methodology has a highly flexible development path. It uses the some of the industries' best practices. These are known as six best practices of RUP methodology.



The 6 RUP Best Practices

1. Develop Iteratively

The software requirements specification (SRS) keeps on evolving throughout the development process and loops are created to add them without affecting the cost of development.

2. Manage Requirements

The business requirements documentation and project management requirements need to be gathered properly from the user in order to reach the targeted goal.

3. Use Components

The components of large project which are already tested and are in use can be conveniently used in other projects. This reuse of components reduces the production time.

4. Model Visually

Use of Unified modeling language (UML) facilitates the analysis and design of various components. Diagrams and models are used to represent various components and their interactions.

5. Verify Quality

Testing and implementing effective project quality management should be a major part of each and every phase of the project from initiation to delivery (aka the project management life cycle).

6. Control Changes

Synchronization of various parts of the system becomes all the more challenging when the parts are being developed by various teams working from different geographic locations on different development platforms. Hence special care should be taken in this direction so that the changes can be controlled.