Introduction to Software and Software Engineering
UNIT -1
Subject: Software Engineering (2160701)
Semester: 6th (BY)

1) Explain evolving role of software.

- Now days, software plays a major role with dual activity. It is a product like a vehicle.
- As a product, it delivers the computing potential embodied by computer hardware or a network of computers that are accessible by local hardware.
- Whether the product or software resides within a mobile phone or operates inside a mainframe computer, software is an information transformer likely to produce, manage, acquire, modify, display or transmit the information.

- The software useful for following things.

    - Provides good product with useful information.
    - Transforms the data so that it can be more useful in a local context.
    - Manages business information to enhance competitiveness.
    - Provides a gateway to worldwide networks like internet.

- The role of computer software has undergone significant change over a time span of little more than 50 years.
- Today, huge software industry has become dominant factor.
- When modern computer system is built it is necessary to replied following questions,

    - Why does it take so long to get software finished?
    - Why development cost is high?
    - Why can't we find all the errors before we deliver software to customer?
    - Why we spent so much time and effort in existing programs?

2) What is Software? Explain Software characteristics.

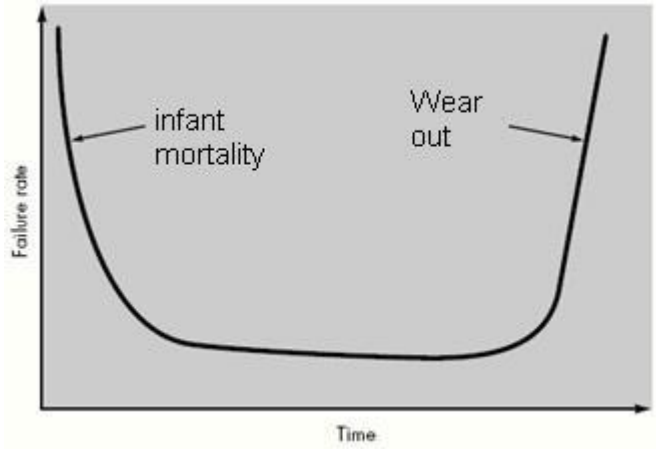
- In 1970, less than 1 percent of the public could have defined what "Computer Software" meant. Today most professionals and many members of the public at large feel that they understand but do they?
- A text book definition of software defined it as follows.
- Software is (1) instructions (computer programs) that when executed provide desired function and performance, (2) data structures that enable the programs to adequately manipulate information, and (3) documents that describe the operation and use of the programs.

Software characteristics.

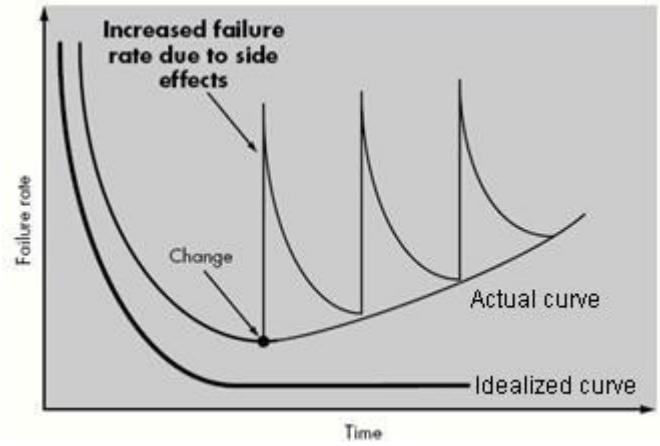
2.1 Software is developed or engineered; it is not manufactured in the classical sense.

- Software development and hardware development are two different activities.
- Quality problems that occur in hardware manufacturing cannot be removed easily on the other hand it can remove easily in software.
- Both the activities are dependent on people, but the relationship between people is totally varying.
- Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects.

2.2 Software doesn't wear out



- Failure rate for hardware is shown in the Figure which often called it as “bathtub curve”. As shown over here initially failure rate is high due to manufacturing defects these defects are corrected and the failure rate drops to a steady-state level for some period of time.
- As time passes, however, the failure rate rises again as hardware components suffer from the environmental factors.
- Stated simply, software is not affected by environment factors. In theory, therefore, the failure rate curve for software should take the form of the “idealized curve” shown in the figure.



- During life cycle of software if any change is made some errors will introduced. This will increase failure rate.
- Before the curve can return to original steady state another change is requested and again failure rate becomes high.

2.3 Although the industry is moving toward component- based assembly, most software continues to be custom built.

- In software industry most of the software are built according to the client’s requirement however now industry slowly moving towards the component based assembly where components of previous project used.
- In hardware industry reusable components like Integrated circuit used frequently which saves developers effort.

3) Explain Software applications domain. OR Explain Software applications.

Software application domain described below.

3.1 System software:

- A collection of programs written to service other programs are called system software.
- Examples are compilers, editors, and file management utilities.

3.2 Real-time software:

- Software that monitors, analyzes and controls real-world events as they occur is called real time.
- Elements of real-time software include a data gathering component that collects and formats information from an external environment.

3.3 Business software:

- The largest single software application area is Business information processing.
- systems like payroll, accounts receivable or payable, inventory are example of business software

3.4 Engineering and scientific software:

- Engineering and scientific software have been characterized by "number crunching" algorithms. Applications range from astronomy to volcano logy, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

3.5 Embedded software:

- Embedded software is a type of software that is built into hardware systems. This software is typically designed to perform one specific function.
- Embedded software resides in read-only memory. It is used to control products and systems for the consumer and industrial markets.
- Embedded software can perform very limited functions such as keypad control for a microwave oven or provide significant function and control capability like digital functions in an automobile such as fuel control, dashboard displays, and braking systems.

3.6 Personal computer software:

- The personal computer software is playing major role in the software market.
- The sample applications are word processing, spreadsheets, computer graphics, multimedia, entertainment

3.7 Web-based software:

- The Web pages retrieved by a browser are software that incorporates executable instructions (e.g., CGI, HTML, Perl, or Java), and data (e.g., hypertext and a variety of visual and audio formats).
- In essence, the network becomes a massive computer providing an almost unlimited software resource that can be accessed by anyone with a modem.

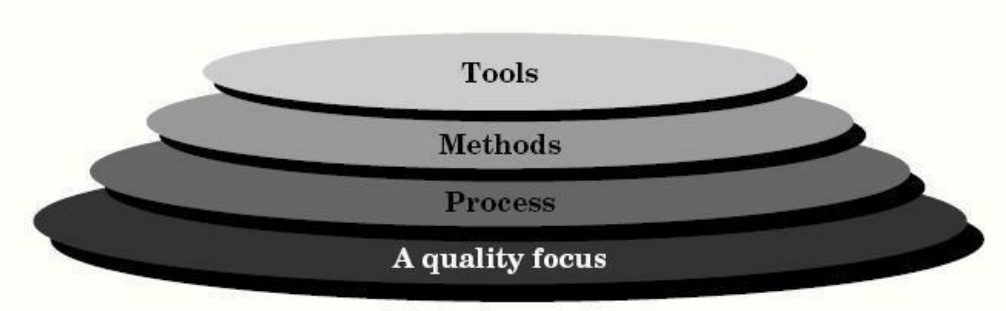
3.8 Artificial intelligence software:

- Artificial intelligence (AI) software makes use of non numerical algorithms to solve complex problems

that are not responsible to computation or straightforward analysis.

- Expert systems, also called knowledge based systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing are representative of applications within this category.

4) Draw and explain Software Engineering layers. OR Layer Technology.



4.1 A Quality Focus :

- Software engineering is a layered technology. Any engineering approach must rest on an organizational commitment to quality .Total quality management will ultimately useful for maintaining quality.

4.2 Process :

- The foundation for software engineering is the process layer. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
- Process defines a framework for a set of key process areas (KPAs) that must be established for effective delivery of software engineering technology.
- The key process areas form the basis for management control of software projects.

4.3 Methods :

- In method layer the actual method of implementation is carried out with the help of requirement analysis, designing, coding, using desired programming constructs and testing.

4.4 Tools :

- Software engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering, is established.
- CASE combines software, hardware, and a software engineering database (a repository containing important information about analysis, design, program construction, and testing) to create a software engineering environment analogous to CAD/CAE (computer-aided design/engineering) for hardware.

5) Explain Process Framework or Draw Common Process Framework.

- Software Process is the total set of software engineering activities necessary to develop and maintain software products.
- It can be characterized as shown in the figure below.
- A *Common Process Framework* is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size and complexity.

- A number of task sets each one is collection of software engineering work tasks, project milestones, software work products and deliverables, and quality assurance points.

- The following generic framework activities are used generally in projects:

  1. Communication:

  Communication play very important role to collect customer requirement. It is heart of software. By communicating with customer requirement gathering is done.

  2. Planning

  It establishes plan for software engineering work. It describes technical task to be conducted, the required resources, work product to be produced etc.

  3. Modeling:

  It contains phase of analysis and design. It creates model that can be understood by customer and developer easily. So that they can have pre idea about the design of product. Analysis encompasses on a set of tasks that lead to create analysis model.

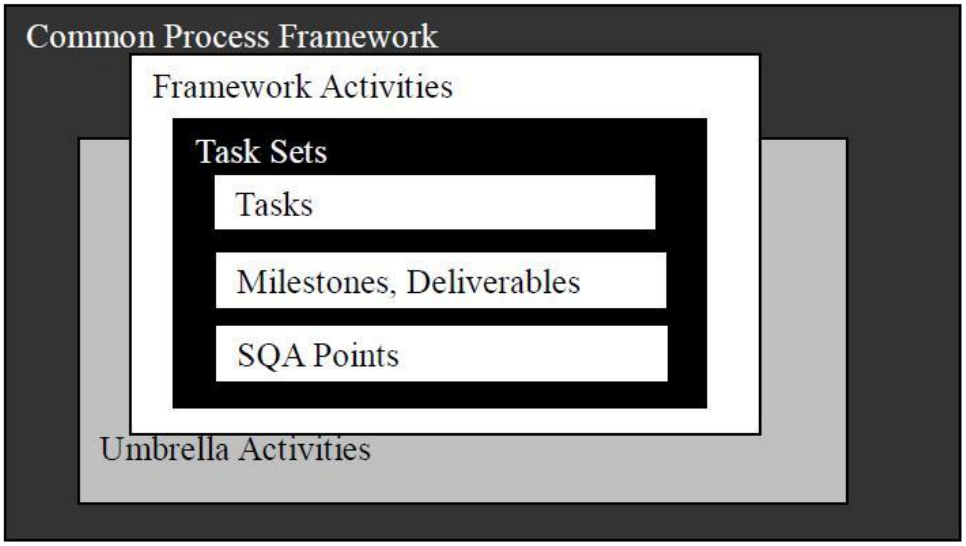
  4. Construction:

  In this phase complete implementation of software is done then all types of testing is performed like black box testing and white box testing etc.

  5. Deployment:

  Whenever product is ready then it will be deployed to the customer either partially or completely. Then customer provides feedback based on evaluation.

- These five generic framework activities can be used during the development of small programs, the creation of large application. The details of software will be very different but the framework activities remain same.
- The framework described in generic view is complemented by a number of umbrella activities. Which is described over here:



Umbrella/Common process framework figure

Umbrella Activities:

1) Software project tracking and control:
It allows the software team to assess progress against the project plan. And take necessary action to maintain a schedule.

2) Risk Management:
It defines risks that may affect the outcome of the project or the quality of the product.

3) Software quality assurance:
It defines and conducts the activities to ensure the software quality.

4) Formal Technical Reviews:
It is necessary to uncover or remove errors before they are propagated to the next action or activity.

5) Software Configuration Management:
It manages the effects of the change throughout the software process.

6) Re-usability Management:
It defines criteria for work product reuse and establishes mechanism to achieve reusable components.

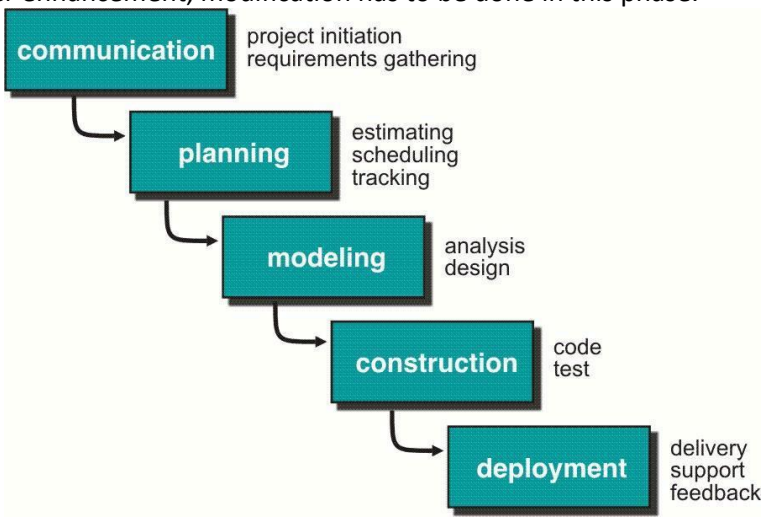
7) Work product Preparation and Production:
It will create work products such as models, documents, logs, forms etc.

8) Measurement:
It defines and collects process, project, and product measures and assists the team to deliver product as per customer's requirements.

- Thus umbrella activities are applied throughout the software process and it is necessary for success.

6) Explain Waterfall model or Linear Sequential model or classic life cycle.

- The waterfall model is also called it as 'linear-sequential model' or 'classic life cycle model'
- The waterfall model proceeds from one phase to the next in a sequential manner.
- By communicating with the customers requirement gathering is complied it will proceed to planning phase. Once design phase is completed then implementation of that design is made by coders in appropriate coding language which is best suited to design.
- Once code is finalized then through testing of code is begin. All testing activities are done in this phase. After testing, the application is being ready to deploy in actual server environment. This phase is implementation phase of actual product.
- Once Application deployed successfully, we can easily modified the application in maintenance phase. The entire customer enhancement, modification has to be done in this phase.



Advantages:

- Provides a disciplined and structured approach which makes it easy to keep projects under control.
- Simple to implement.
- For implementation of small system waterfall model is useful.

Disadvantages:

- Sometimes it is difficult to maintain sequential flow in the project.
- Requirement analysis is done initially and sometimes it is not possible to state all the requirements explicitly in the beginning. This cause difficulty in the project.
- The deliverable (ideas into working software) is a one time.
- Linear nature of waterfall model induces blocking states, because certain tasks may be dependant on some previous tasks.

7) Prototyping Model OR
What is software prototyping? Explain its significance in software engineering with example.

- The Prototyping Model is a systems development method in which a prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.
- This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.
- There are several steps in the Prototyping Model:

i. Preliminary design is created for the new system.

ii. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

iii. The users thoroughly evaluate the first prototype, noting its strengths and weaknesses, what needs to be added, and what should to be removed. The developer collects and analyzes the remarks from the users.

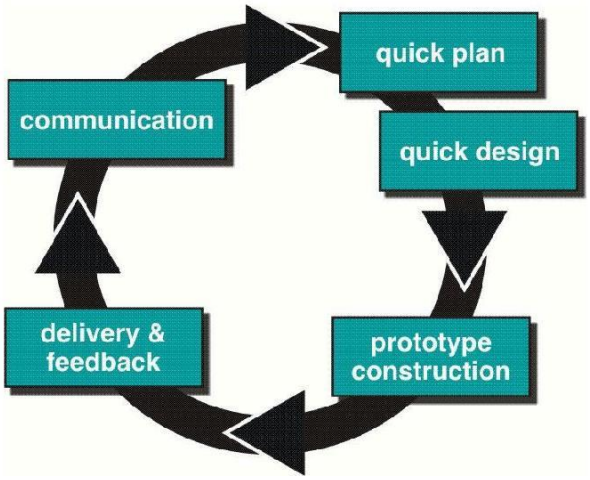
iv. The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed.

v. The second prototype is evaluated in the same manner as was the first prototype.

vi. The preceding steps are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired.

vii. The final system is constructed, based on the final prototype.

viii. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.



Advantages:

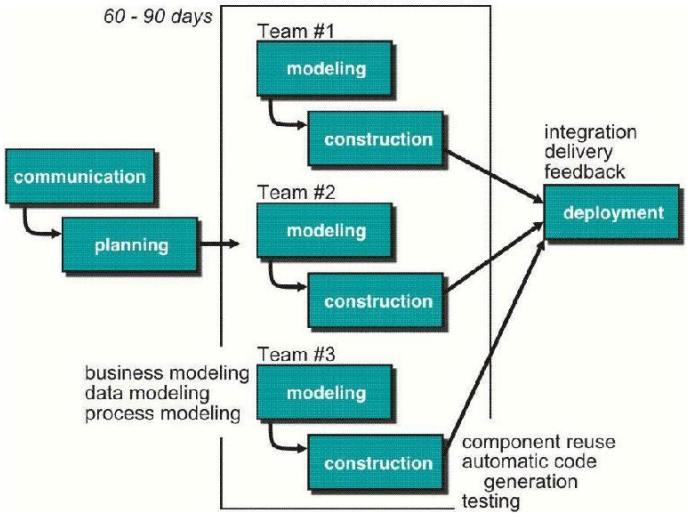
- Reduced time and costs
- Improved and increased user involvement.

Disadvantages:

- Insufficient analysis.
- User confusion of prototype and finished system
- Developer misunderstanding of user objectives
- Expense and time of implementing prototyping

8) Rapid application development (RAD) Model.

- Rapid application development (RAD) is an incremental software development process model that has extremely short development cycle.
- The RAD model is a "high-speed" adaptation of the waterfall model in which rapid development is achieved by using component-based construction. If requirements are well understood and project scope is properly defined then developers implement this model in extremely short time.
- Communication works to understand the business problem and information characteristics that the software must accommodate.
- Planning is essential because multiple software teams work in parallel on different system functions.
- Modeling include three activities business modeling, data modeling and process modeling and it establishes design representations that serve as the basis for RAD's construction activity.
- Construction emphasizes the use of pre-existing software components and the application of automatic code generation.
- Finally, deployment establishes a basis for subsequent iterations, if required.
- Obviously, the time constraints imposed on a RAD project demand "scalable scope".
- If a business application can be modularized in a way that enables each major function to be completed in less than three months then software team can make use of RAD model.
- Each major function can be addressed by a separate RAD team and then integrated to form a whole.
- The RAD process model is illustrated in Figure

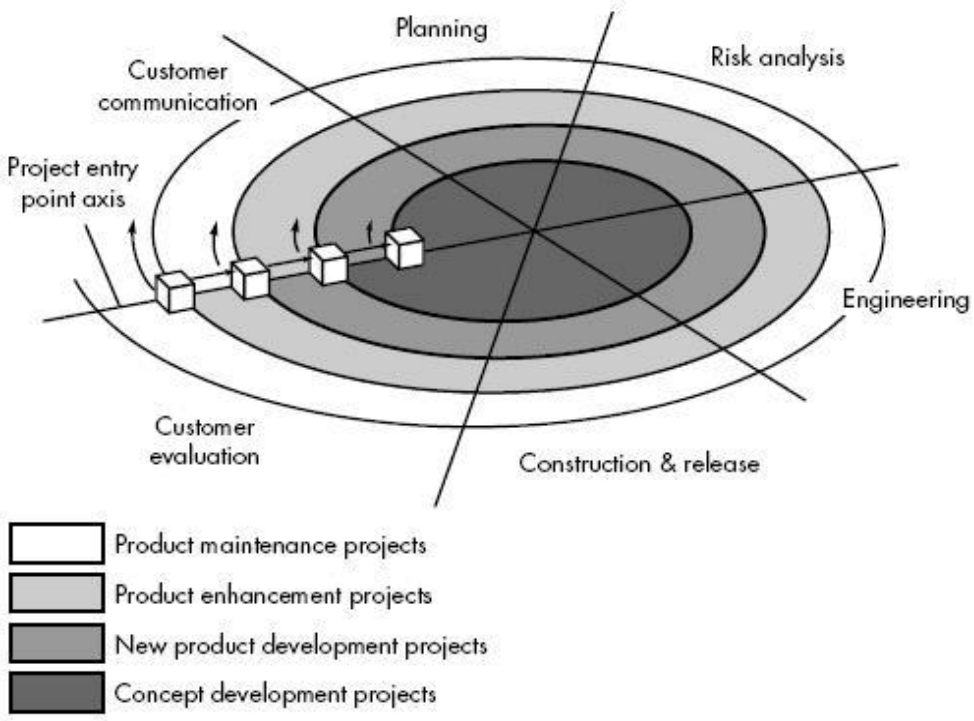


Drawbacks :

- For large but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
- RAD requires developers and customers who are committed to the rapid-fire activities necessary to get a system complete in a much abbreviated time frame.
- If commitment is lacking from either constituency, RAD projects will fail.
- If a system cannot be properly modularized, building the components necessary for RAD will be problematic.
- If high performance is an issue and performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work.
- RAD is not appropriate when technical risks are high.

9) 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 linear sequential model. In Spiral Model, software is developed in a series of incremental releases.
- During early iterations, the incremental release might be paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.
- The spiral model is divided into a number of framework activities, also called task *regions.* There may be between three and six task regions. Figure below depicts a spiral model that contains six task regions.



- Customer communication required to establish effective communication between developer and customer.
- Planning required to define resources, timelines, and other project-related information.
- Risk analysis required to assess both technical and management risks.
- Engineering required to build one or more representations of the application.
- Construction and release required to construct, test, install, and provide user support (e.g., documentation and training).
- Customer evaluation required to obtain customer feedback based on evaluation of the software representations created during the engineering stage and implemented during the installation stage.

- The software engineering team moves around the spiral in a clockwise direction it begins at the centre and later on it will generate more useful versions of software.
- Each pass through the planning region results in adjustments to the project plan. Cost and schedule are adjusted based on feedback derived from customer evaluation.The planned number of iterations required to complete the software.
- Each iterations are explained below.

Concept Development Project:

- Start at the core and continues for multiple iterations until it is complete.
- If concept is developed into an actual product, the process proceeds outward on the spiral.

New Product Development Project:

- New product will evolve through a number of iterations around the spiral.
- Later, a circuit around spiral might be used to represent a “Product Enhancement Project”

Product Enhancement Project:

- There are times when process is dormant or software team not developing new things but change is initiated, process start at appropriate entry point.

Product Maintenance Concept:

- It is the Longest phase in which software maintained by software team.

Advantages:

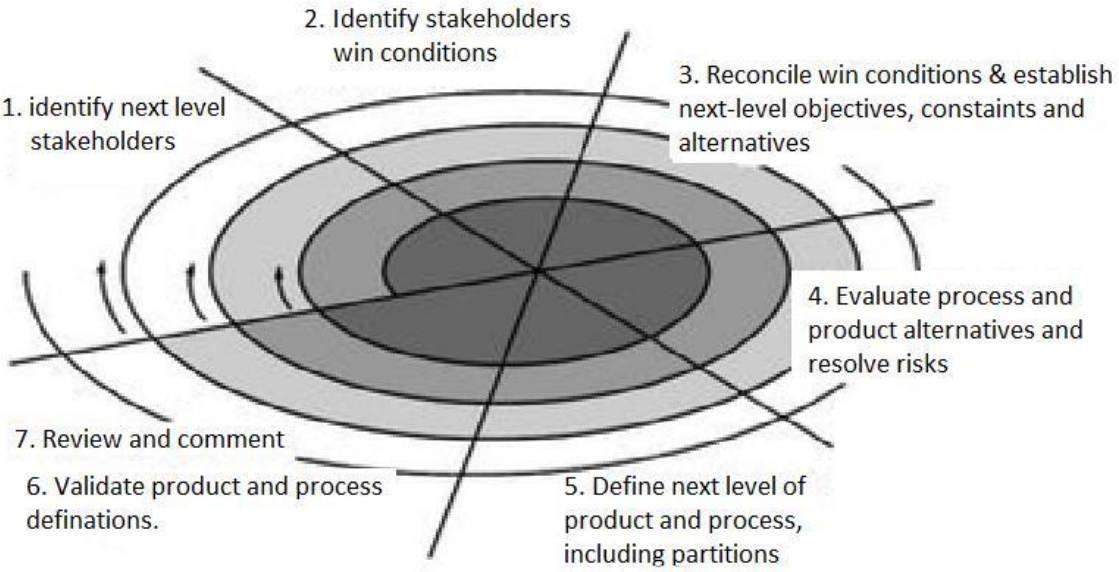
- High amount of risk analysis.
- Good for large and mission-critical projects.
- Software is produced early in the software life cycle.

Disadvantages:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects

10) Explain Winwin spiral model

- Winwin spiral model also need communication phase initially in the project.
- In reality, the customer and the developer enter into a process of negotiation Successful negotiation occurs when both sides WIN.
- Customer and developer look for a "win-win" result. That is, the customer wins by getting the system or product that satisfies the majority of the customer's needs and the developer wins by working to realistic and achievable budgets and deadlines.
- Boehm's WINWIN spiral model defines a set of negotiation activities at the beginning of each pass around the spiral.
- Rather than a single customer communication activity, the following activities are defined:



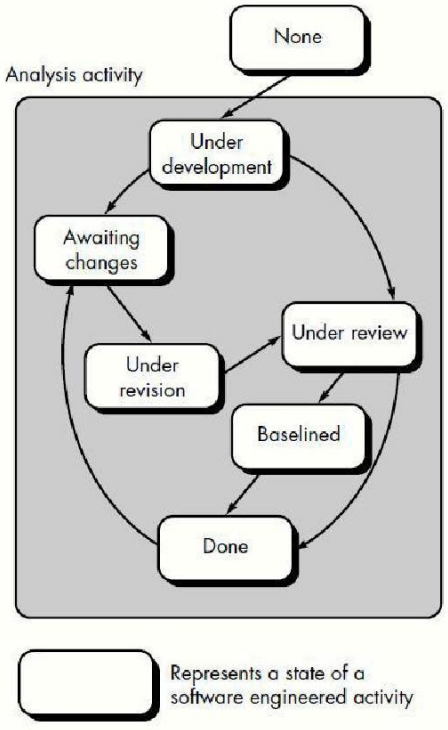
- Identification of the system stakeholders. That is the people on organisation that have direct business interest in the product to be built and will be rewarded for a successful outcome or criticised if the effort fails.
- Determination of the stakeholder's "win conditions"
- Negotiations of the stakeholder's win conditions to reconcile them into a set of win-win conditions for all concerned (including the software project team).
- Successful completion of these initial steps achieves a win-win result, which becomes the key criterion for proceeding to software and system definition.
- In addition to the early emphasis placed on the win-win condition, the model also introduces three process milestones (anchor points), which help establish the completion of one cycle around the spiral and provide the decision milestones before the software project proceeds. These are,
- Life Cycle Objectives (LCO) which Defines a set of objectives for each major software activity (e.g. a set of objectives associated with the definition of top level product requirements)
- Life Cycle Architecture (LCA) which Establishes the objectives that must be met as the as the software architecture is defined.
- Initial Operational Capability (IOC) which represents a set of objectives associated with the preparation of the software for installation/distribution, site preparations prior to installations, and assistance required by all parties that will use or support the software.

Advantages:

- Faster software production facilitated through collaborative involvement of the relevant stakeholders.
- Cheaper software via rework and maintenance reductions

11) Concurrent Development Model.

- This model is also called it as concurrent engineering model. In this model the framework activities are represented as series of tasks.
- For example the modeling activity can performed in various stages.
- This states contain various activities or tasks.
- For example : Modeling activity can be initially in under development state. then when certain processes or activities are going on it might be in awaiting changes state.
- Some reviews or revisions might be carried out on the developed or potentially developed software product.
- Hence under review or under revision might be some states.
- Finally software product goes in done state.



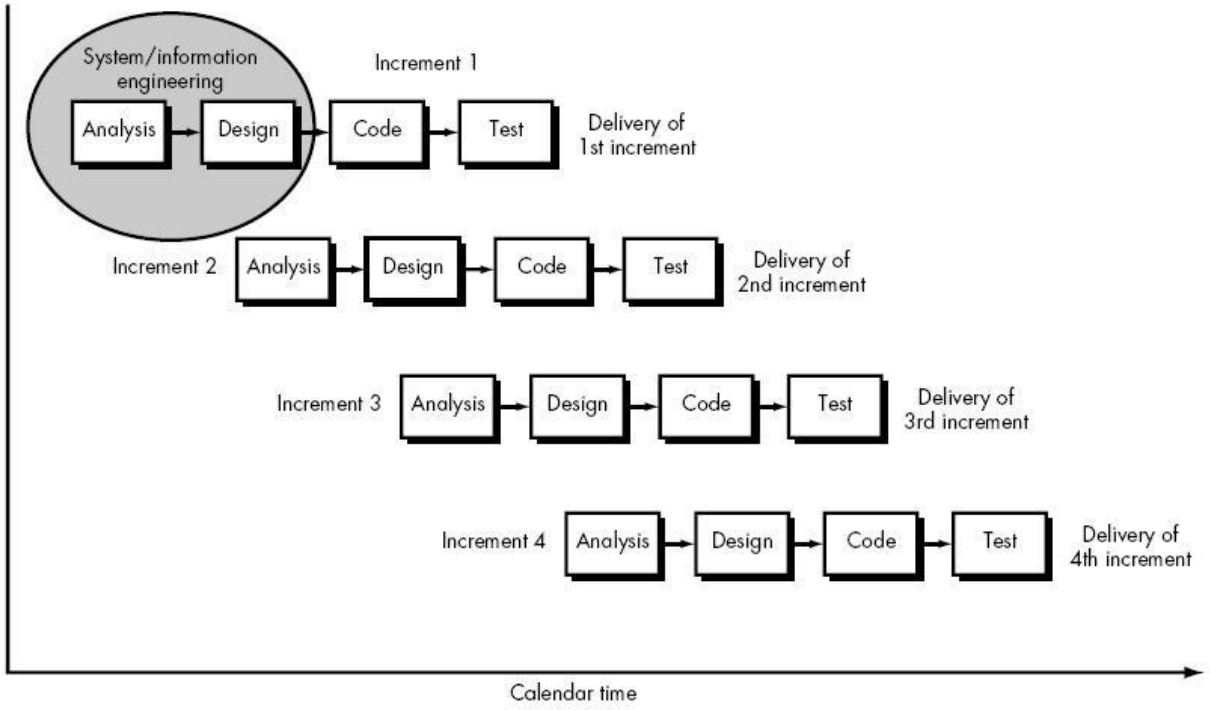
- Sometimes some inconsistencies in analysis model may trigger some changes in the existing product.Then implementing those changes product has to undergo
  awaiting changes state.

Advantages:

- All type of software development can be done using this model.
- This model provides accurate picture of current state if project.
- Each activity carried out concurrently hence it is efficient model.

12) Incremental model.

- Incremental model is used when software requirements are well defined and there is need to provide set of software functionality quickly.
- Incremental model combines linear and parallel flows. Each linear sequence produces deliverable "increments" of software.
- For example, word processing software is developed using this model then in first increment basic file management and editing functionalities delivered.
- In second increment documentation production ability is delivered.
- In third increment spelling checking ability is delivered.
- Here in this model first increment is often called it as "core product". Then additional features will be added in the later increments.
- Incremental model is particularly useful when staffing is unavailable for complete implementation by the business deadline.
- If core product is well received, then additional staff can be added to implement the next increment if required.



Advantages

- Early increments can be developed with few people. It combines iterative nature of prototyping model and linear nature of Linear Sequential Model.
- There is less number of people required to implement this model.
- Easy to add quality in this model.
- The system can be designed in such a manner that it can be delivered into pieces.
- Increments are developed one after the other after delivering increment feedback has been received from the user.
- Since each increment is simpler than the original system, it is easier to predict resources needed to accomplish the development task within acceptable accuracy bounds.
- Increments can be planned to manage technical risks.

Disadvantages

- Resulting cost may exceed the cost of the organization.
- As additional functionality is added to the product, problems may arise related to system architecture which were not evident in earlier prototypes.

13) Myth in software engineering and its various types.

Software Myths

- Software Myths consist beliefs about software and the process used to build it this can be traced to the earliest days of computing. Myths have a number of attributes that have made them insidious. For instance, myths appear to be reasonable statements of fact, they have an intuitive feel, and they are often promulgated by experienced practitioners who "know the score".

Management Myths

- Managers with software responsibility, like managers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping, and improve quality. Like a drowning person who grasps at a straw, a software manager often grasps at belief in a software myth, If the Belief will lessen the pressure.
  - **Myth:** We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?
  - **Reality:** The book of standards may very well exist, but is it used? - Are software practitioners aware of its existence? - Does it reflect modern software engineering practice? - Is it complete? Is it adaptable? - Is it streamlined to improve time to delivery while still maintaining a focus on Quality? In many cases, the answer to these entire question is no.
  - **Myth:** If we get behind schedule, we can add more programmers and catch up (sometimes called the Mongolian horde concept)
  - **Reality:** Software development is not a mechanistic process like manufacturing. In the words of Brooks "Adding people to a late software project makes it later." At first, this statement may seem counterintuitive. However, as new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort
  - **Myth:** If we decide to outsource the software project to a third party, I can just relax and let that firm build it.
  - **Reality:** If an organization does not understand how to manage and control software project internally, it will invariably struggle when it out sources software project.

Customer Myths

- A customer who requests computer software may be a person at the next desk, a technical group down the hall, the marketing /sales department, or an outside company that has requested software under contract. In many cases, the customer believes myths about software because software managers and practitioners do little to correct misinformation. Myths led to false expectations and ultimately, dissatisfaction with the developers.
  - **Myth:** A general statement of objectives is sufficient to begin writing programs we can fill in details later.
  - **Reality:** Although a comprehensive and stable statement of requirements is not always possible, an ambiguous statement of objectives is a recipe for disaster. Unambiguous requirements are developed only through effective and continuous communication between customer and developer.

Practitioner's Myths

- Myths that are still believed by software practitioners have been encouraged by decades of programming culture. During the early days of software, programming was viewed as an art form. Old ways and attitudes die-hard.
  - **Myth: Once we write a program and get it to work, our job is done.**
  - **Reality:** Someone once said," The sooner you begin writing code, the longer it will take you to get done". The industry data indicates that the majority of effort expended on a program will be after the program has been delivered to the customer for the first time.
  - **Myth:** Until I get the program running I have no way of assessing its quality.
  - **Reality:** One of the most effective software quality assurance mechanisms is the formal technical review. These can be undertaken long before the program is running.
  - **Myth:** The only deliverable for a successful project is the working program.
  - **Reality:** A working program is only one of the elements of the project. Other elements include: project plans, requirements specifications, system designs, test specifications, support documentation etc. Documentation forms the foundation for successful development and, more importantly, provides the foundation for the software maintenance task.