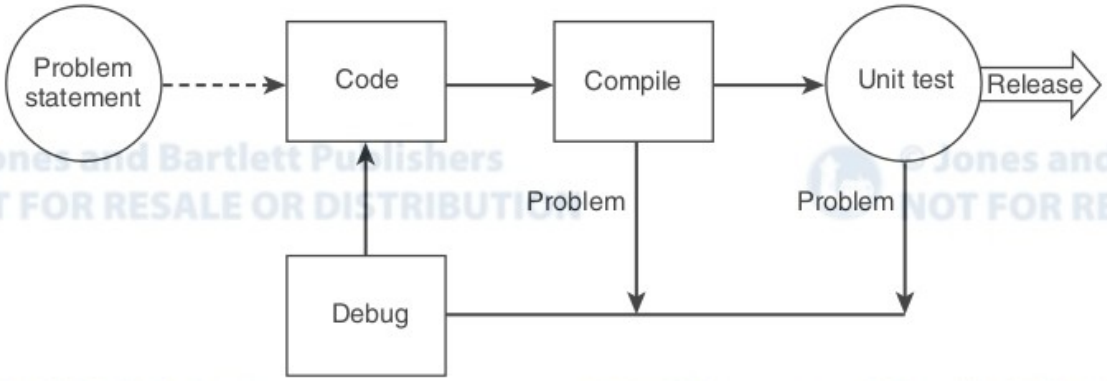
Assignment 1:

Q.1 What do you understand by the terms Software Process and Software Process Model?[4 marks]

- 1.1. Software Process:
  - why do we follow software processes:
  - For:
    - Repeatibility
    - Predictability
    - Traceability
    - Improved quality through standardisation
    - Continuous improvement
    - Enables training
    - Builds confidence



- Fig:a simple sofrware process.

- The software process defines an ordered set of steps to produce a high quality software system.
- It is a systematically designed method of developing and maintaining a software system through its life cycle.
- It is the way software development is organised, managed, measured, supported and improved, independently of the support techniques used in the development.
- Effective management of Software process is the key to succesful delivery (on time, on budget, with the expected quality) of software .
- We need to follow software processes to reduce risk in software development.

1.2.Software Process Model:

- A Software Process Model is a simplified representation of a software process that is presented from a specific perspective.

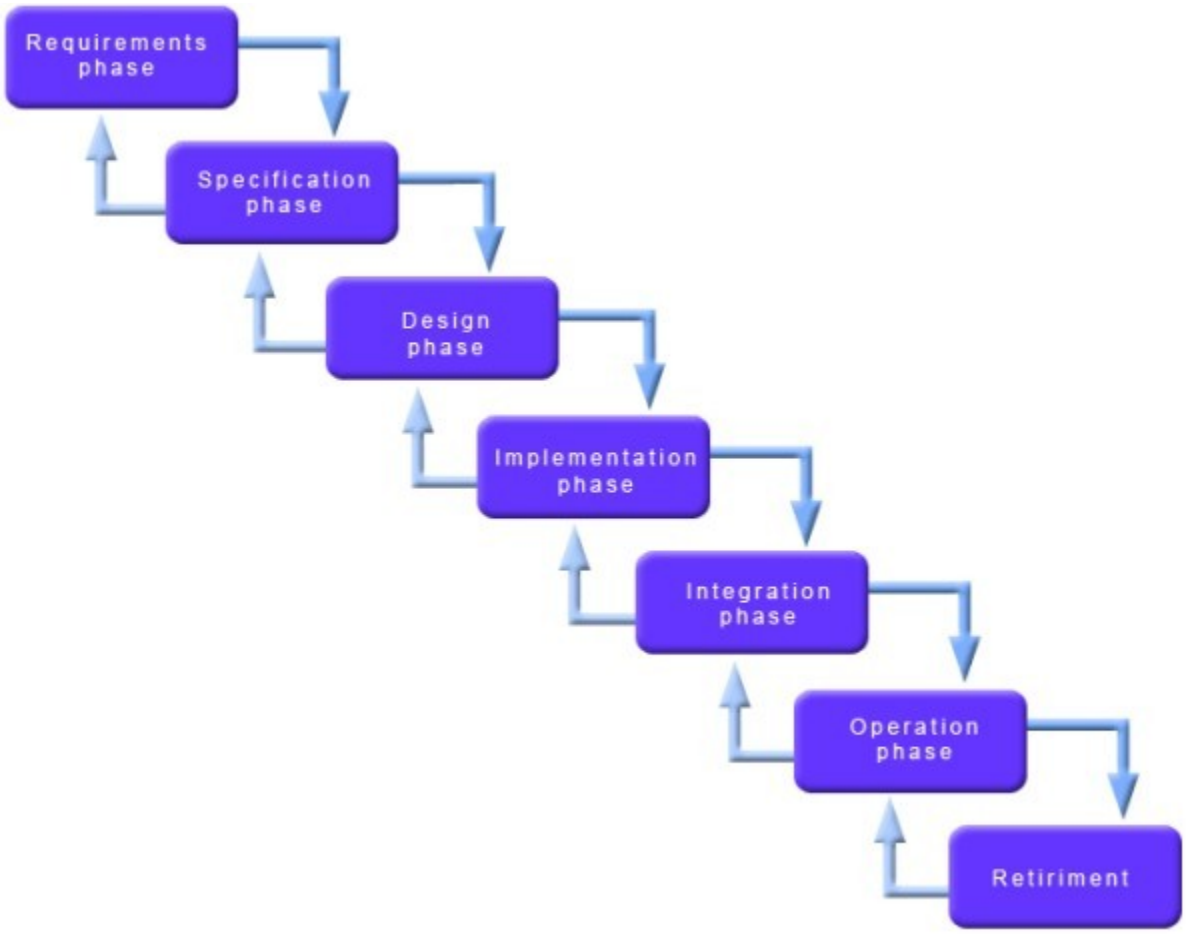
- It describes the creation of software development process.
- It is an abstract representation of a process. it represents a description of a process from a particular perspective.
- The goal of a software process model is to provide guidance for systematically coordinating and controlling the tasks that must be performed in order to achieve the end product and the project objectives.
- A process model defines the following:
    - A set of tasks that need to be performed.
    - The input to and output from each task.
    - The preconditions and postconditions for each task.
    - The sequence and flow of these tasks.

Q.2 Explain briefly each of the following software life cycle models with a suitable diagram.[5x7=35 marks]

2.1 Linear Sequential Model/Water fall Model
2.2 Prototyping Model
2.3 Rapid Application Development Model
2.4 Evolutionary Software Process Models
2.5 Incremental Model
2.6 Spiral Model
2.7 Concurrent Development Model

2.1.Linear sequential model(Water fall model):

- Sometimes called the classic life cycle or the waterfall model.
- Waterfall model is the most well known software lifecycle development model. It is very simple to understand and use. Each next phase in this model must begin only after the previous phase is over.
- Waterfall software development model may be applicable to projects where:
    1. Software requirements clearly defined and known
    2. Software development technologies and tools is well known
    3. New version of the existing software system is created.
- The linear sequential model suggests a systematic, sequential approaches to software development that begins at the system level and progresses through analysis, design, coding, testing, and support.
- A product is delivered after the linear sequence is complete.



Requirements analysis and definition. The system services, constraints and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

System and software design. The system design process partitions the requirements to either hardware or software systems. It establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

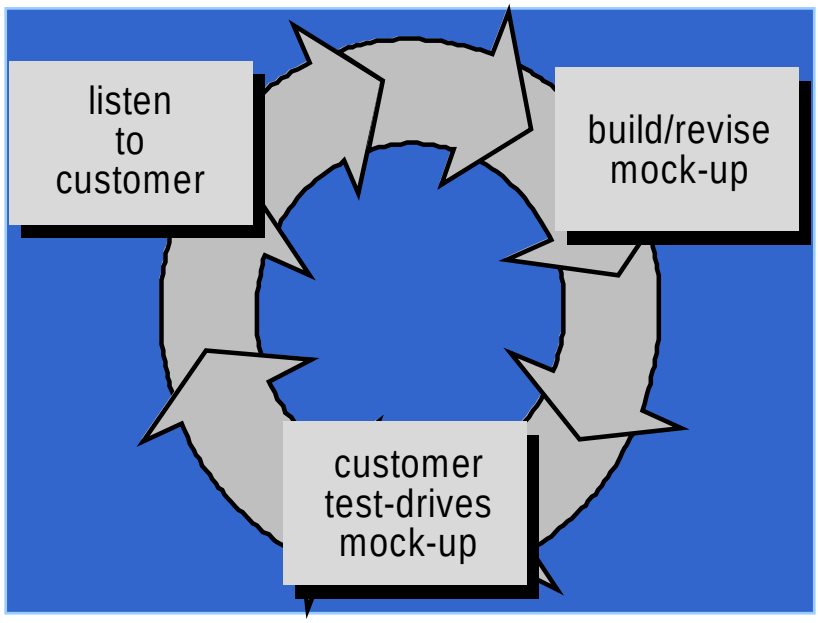
Implementation and unit testing. During this stage, the software design is realized as a set of programs or program unit. Unit testing involves verifying that each unit meets its specification.

Integration and system testing. The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

Operation and maintenance. Normally this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which

were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

2.2.The Prototyping Model



- Here, a prototype is made first and based on it final product is developed. A prototype is a model or a program which is not based on strict planning, but is an early approximation of the final product or software system. A prototype acts as a sample to test the process.
- The prototyping of software is an approach to software development which challenges conventional wisdom. The features of prototyping are that:
    - There is no perfect knowledge of the application before implementation.
    - Evolution is a natural part of the life of software.
    - Users of the software are involved during all stages of software development.
    - The prototype itself is the primary communication mechanism between users and developers.
- Prototyping has long been used by software developers to test software on a small scale. Prototyping on a larger scale is far less common, however.
- In the more revolutionary form of prototyping, evolutionary prototyping, software is developed and delivered incrementally to the users.
- Functions are added or deleted as the software evolves,towards a satisfactory form.
- Generally in this model developers listen to customer, design a prototype and then the prototype is tested.
- The procedure is repeated until all the requirements of the system are identified and a complex system can be designed for the system.

- Prototyping serves as a mechanism for identifying **software requirements**.
- It begins with **requirements gathering**, and then a **quick design** that focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approach and output formats). The quick design leads to the construction of a **prototype**.
- With prototyping model, users get a feel for the actual system, and developers get to build something immediately.
- A prototype may be discarded because it is too slow, too big, too awkward in use, not running with the right programming language, not on the right platform, …

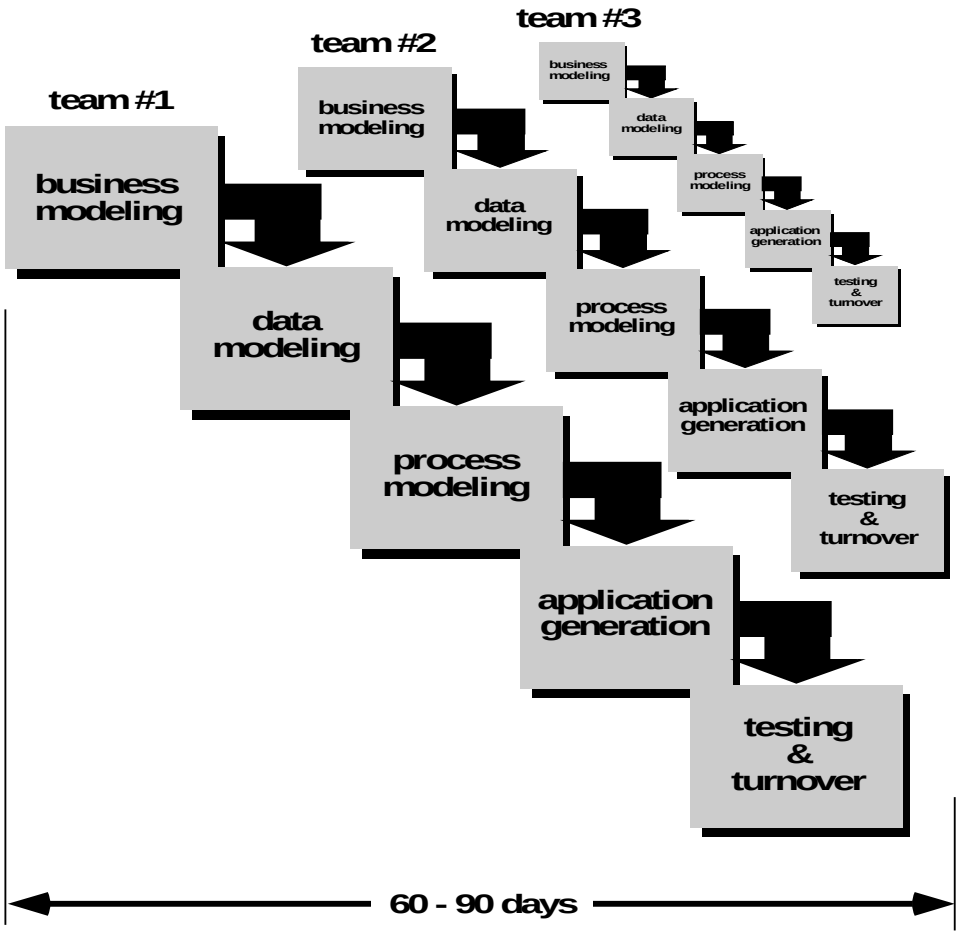
Suitable scenario:

- General objectives of software are defined.
- Detail requirements of input, processing and output are not identified.
- Efficiency of algorithms is not sure.
- Adaptability can take place.

The model's objective:

- Defined rules are used in the prototype to identify the detail requirements.

2.3.The RAD (Rapid Application Development) Model



- RAD is an incremental SW process model that emphasizes a short development cycle.
- RAD model is a “high-speed” adaptation of the waterfall model, in which rapid development is achieved by using a component-based construction approach.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within a very short time period.
- 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 3 months, it is a candidate for RAD Each major function can be addressed by a separate RAD team and then integrated to form a whole
- It reuses existing programming components (when possible) or creates reusable components (when necessary).

- RAD assumes the use of 4GT (Fourth Generation Techniques); automated tools are used to facilitate construction of the software.

2.4.Evolutionary Software Process Models

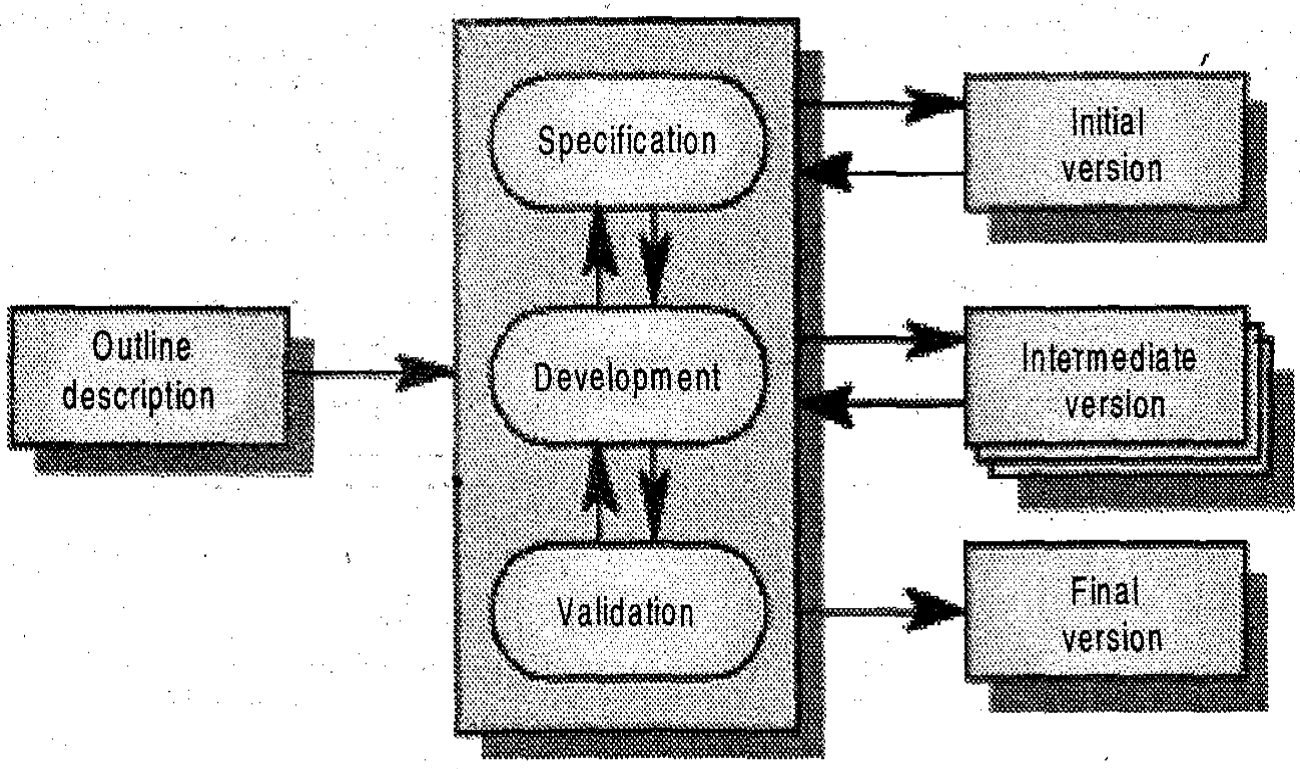


Fig:Evolutionary Model.

The evolutionary model is also called prototyping or simulation model.

- The Evolutionary model shows software development as a series of rises, each

representing a separate loop of the spiral.

- It shows that loops, or releases, tend to overlap each other that make it clear that development work tends to reach a peak, at around the time of the deadline for completion.
- It shows that each prototype or release can take the different amounts of time and efforts to deliver.
- The increments in this model are developed in a sequential way rather than in a parallel.
- There is a normal progression through analyses, design, code, test, implementation and maintenance within the each incremental development cycle.

2.4.1.The Incremental Model

- The incremental model may be viewed as a modification to the waterfall model.
- As software projects increased in size, it was recognized that it is much easier, and sometimes necessary, to develop the software if the large projects are subdivided into smaller components, which may thus be developed incrementally and iteratively.
- In the early days, each component followed a waterfall process model, passing through each step iteratively.
- In the incremental model the components were developed in an overlapping
- fashion, as shown in Fig.below.
- The components all had to be integrated and then tested as a whole in a final system test.
- The incremental model provided a certain amount of risk containment. If any one component ran into trouble, the other components were able to still continue to be developed independently.
- Unless the problem was a universal one, such as the underlying technology being faulty, one problem would not hold up the entire development process.
- This allows thedevelopment team to demonstrate results earlier on in the process and obtain valuable feedback from system users.
- Often, each iteration is actually a mini-Waterfall process with the feedback from one phase providing vital information for the design of the next phase.
- This model delivers software in small but usable pieces, called **increments.**
- Each linear sequential produces a deliverable increment of the software**.**
- The first increment is often a *core product*.
- In general, each increment builds on those that have already been delivered.
- When you encounter a difficult *deadline* that cannot be changed or when *staffing*

is unavailable, the incremental model is a good paradigm to consider.

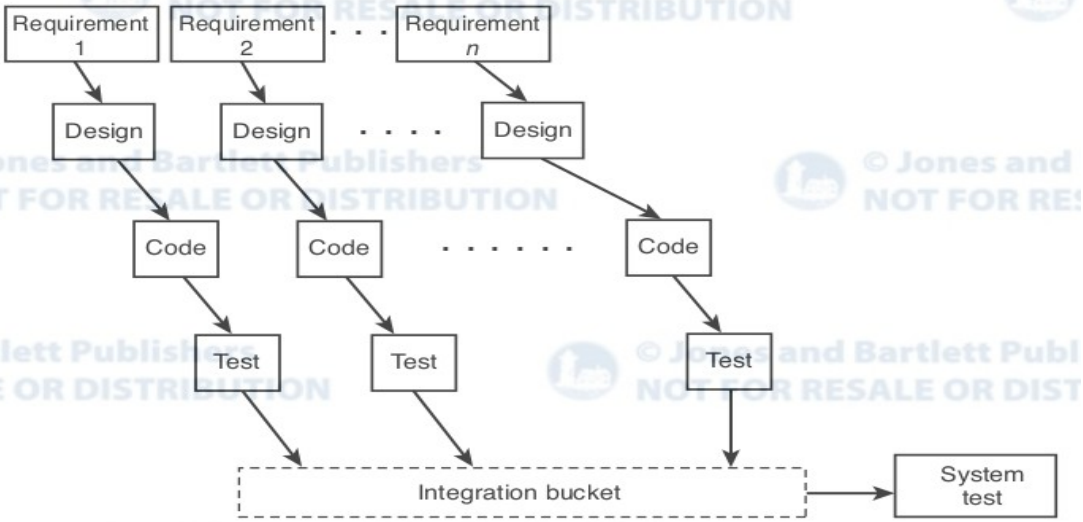
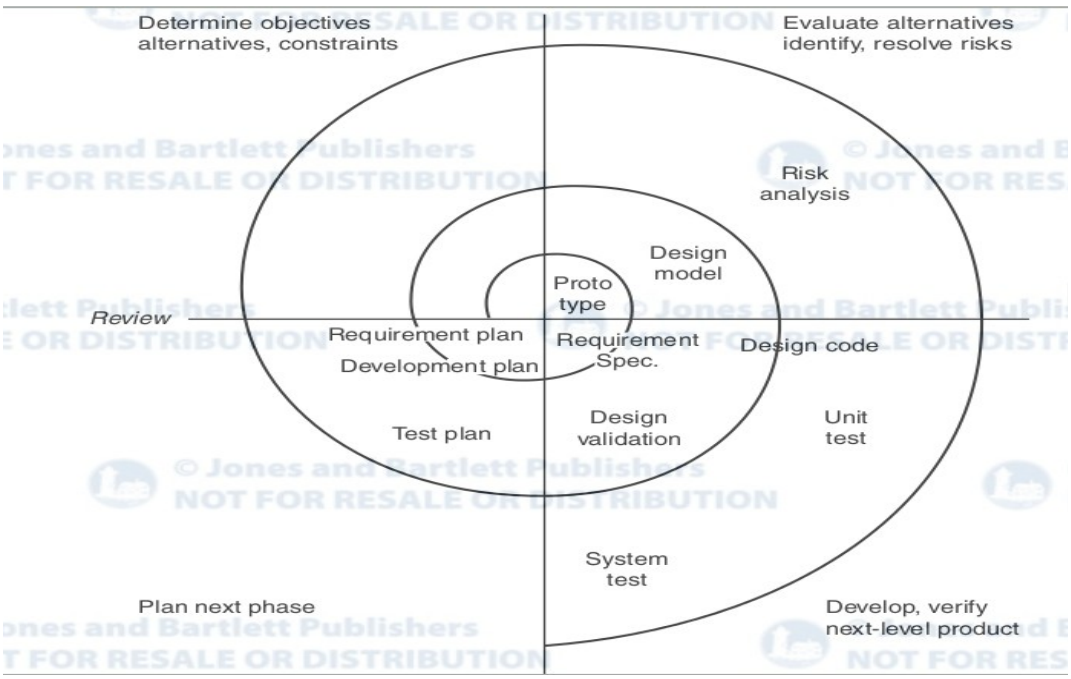


Fig :multiple component incremental model

- The incremental model in Figure above would have individual releases. For example, Requirement 1 would be the core functionality release.
- Other requirements would each depict different deliveries.

2.4.2. The Spiral Model



- With this model, software is developed in a series of incremental releases.
- During early iterations, the incremental release might be a paper model or prototype; during late iterations, increasingly more complete versions of the engineered system are produced.

- A spiral model is divided into a number of framework activities, also called task regions. The spiral model can be adapted to apply throughout the life of computer software. This model is a realistic approach to the development of large-scale system and software.
- This model enables developers to apply the **prototyping** approach at any stage in the evolution of the product.
- The model is thus a risk-driven approach to software process.
- It provides a cyclic approach to incrementally develop the software system while reducing the project risk as the project goes through cycles of development, as illustrated in Figure above.
- The spiral model has four quadrants, and the software project traverses through the quadrants as it is incrementally developed.
- A typical traversal through the four quadrants is as follows:
  1. Identify the objectives, alternatives, or constraints for each cycle of the spiral.
  2. Evaluate the alternatives relative to the objectives and constraints. In performing this step, many of the risks are identified and evaluated.
  3. Depending on the amount of and type of identified risks, develop a prototype, more detailed evaluation, an evolutionary development, or some other step to further reduce the risk of achieving the identified objective.
  On the other hand, if the risk is substantially reduced, the next step may just be a task such as requirements, design, or code.
- 4. Validate the achievement of the objective and plan for the next cycle.

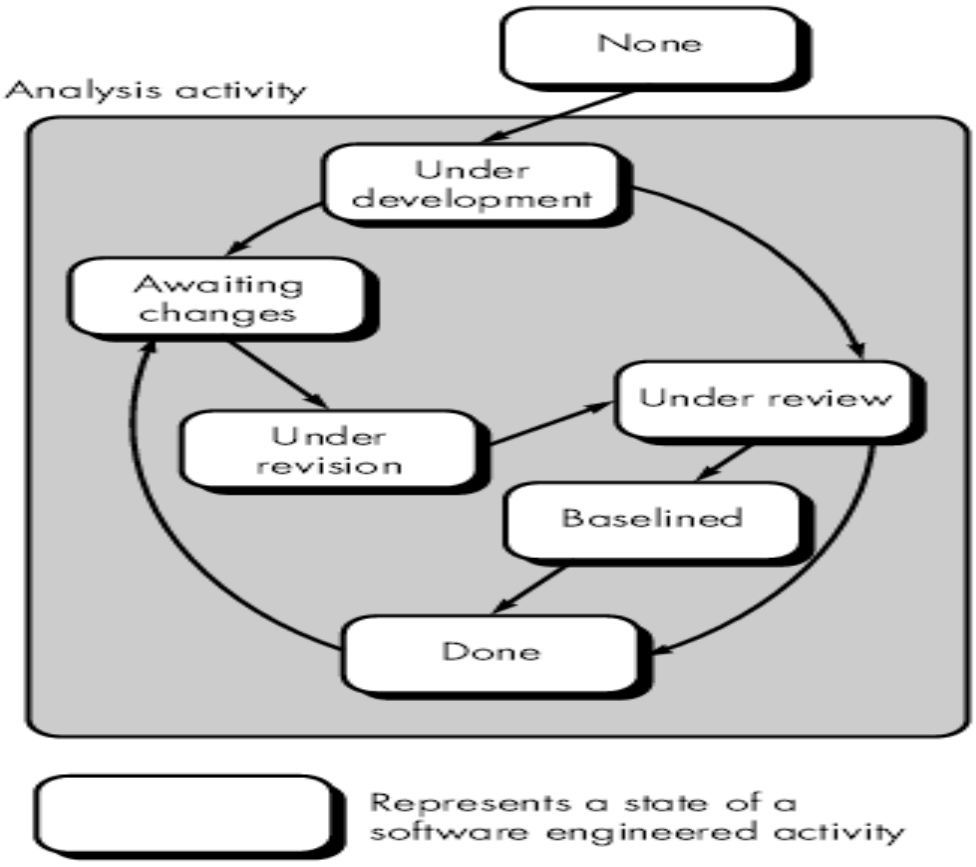
- A product needs several projects to complete. Multiple iterations might occur along the spiral path for each project until the project is completed.

2.4.3 Concurrent Development Model:

- The concurrent process model defines a series of events that will trigger transition from state to state for each of the SW engineering activities, actions or tasks. All activities exist concurrently but reside in different states:
  E. g. The modeling activity which existed in **none** state while initial communication was completed, now makes a transition into **under development**

state. If customer indicates that changes in requirements must be made, modeling activity moves from **under development** state into **awaiting changes** state.

- Is applicable to all types of software development and provides an accurate picture of the current state of project
- Each activity, action, or task on the network exists simultaneously with other activities, actions, or tasks. Events generated at one point in the process network trigger transitions among the states
- The concurrent process model can be represented schematically as a seriesof major technical activities, tasks, and their associated states.



.

- Often used as the paradigm for the development of client/ server applications.

Question 3:Mention some advantages and disadvantages of each model. [3x7=21 marks]

3.1.WATERFALL MODEL:

Advantages:	Disadvantages:
• It is linear and therefore very easy to be implemented-:simple concept. • Required amount of resources are minimal. • A documentation is produced at every stage of waterfall model development. • After every major stage of coding, testing is done to check whether the code running is correct or not. • Clearly divides the problem into distinct phases that may be performed independently. • Natural approach to solving the problem. • Fits well into a contractual setting where each phase is considered a milestone.	• Tester role only happen in the test phase. • A phase should not start until the previous phase is signed off. • Partitioning into the distinct stages of projects is inflexible. • Cannot make changes in middle or any phases after the process is started-Unable to move back to the previous phase. • Not simultaneous. • Can be used only when the requirements are fixed. • Adjusting scope during the life cycle can kill a project • If any mistake happen on middle phases, should start from the scratch. • High amounts of risk and uncertainty.

3.2.THE PROTOTYPING MODEL:

Advantages:	Disadvantages:
• This type of approach of developing the software is used for non-IT-literate people. They usually are not good at	• The start-up cost of building the development team, focused on making prototype, is high.

specifying their requirements, nor can tell properly about what they expect from the software. • Client gets to judges capabilities of developer through prototype,thus developers receive qualifiable user feedback. • It reduces risk of failure, as potential risks can be identified early and mitigation steps can be taken. • Iteration between development team and client provides a very good and conductive environment during project. • Reduces development timeand costs. • Requires user involvement. • Facilitates system implementation since users know what to expect. • Results in higher user satisfaction • Exposes developers to potential future system enhancements.	• Sometimes requirements gathered from client after showing prototype model, may be of no use. That is why, sometimes the prototype model is refered to asas "Throw-away" prototype. • It is a slow process. • Too much involvement of client, is not always preferred by the developer. • Too many changes can disturb the rhythm of the development team. • Can lead to insufficient analysis. • Users expect the performance of the ultimate system to be the same as the prototype. • Developers can become too attached to their prototypes. • Can cause systems to be left unfinished and/or implemented before they are ready. • Sometimes leads to incomplete documentation. • If sophisticated software prototypes (4th GL or CASE Tools) are employed, the time saving benefit of prototyping can be lost.

3.3The RAD (Rapid Application Development) Model

Advantages:	Disadvantages:
• Time to deliver is less. • Changing requirements can be accommodated. • Progress can be measured. • Cycle time can be short with use of powerful RAD tools. • Productivity with fewer people in short	• Management complexity and Resource requirements is more. • Suitable only for systems that are component based and scalable andwhen requirements are well known. • Requires user involvement throughout the life cycle,which isnt always possible

time. • Use of tools and frameworks. • Users/customers own requirements Instills customer confidence that the "right" product is being built. • Provides a good way to determine requirements when there is uncertainty about what is needed	• Suitable for project requiring shorter development times. • Validation may be difficult because the requirements may not be well documented • For large but scalable projects, RAD requires sufficient human resources. • RAD requires developers and customers who are committed to the rapid-fire activities. • Not all types of applications are appropriate for RAD. If a system cannot be properly modularized, building the components necessary for RAD will be problematic. • RAD is not appropriate when technical risks are high.

3.4.Evolutionary Software Process Models

Advantages:	Disadvantages:
• Risk analysis is better. • It supports changing requirements. • Initial Operating time is less. • Better suited for large and mission-critical projects. • During life cycle software is produced early which facilitates customer evaluation and feedback.	• Not suitable for smaller projects. • Management complexity is more. • End of project may not be know which is a risk. • Can be costly to use. • Highly skilled resources are required for risk analysis. • Project's progress is highly dependent upon the risk analysis phase.

3.5.The Incremental Model

Advantages:	Disadvantages:
It is useful when staffing is unavailable for the complete implementation.	• More resources may be required. • Although cost of change is lesser

• Can be implemented with fewer staff people. • If the core product is well received then the additional staff can be added. • Customers can be involved at an early stage. • Each iteration delivers a functionally operational product and thus customers can get to see the working version of the product at each stage.	but it is not very suitable for changing requirements. • More management attention is required. • Each phase of an iteration is rigid with no overlaps. • System architecture or design issues may arise because not all requirements are gathered at beginning for the entire life cycle. • Does not allow iterations within an increment. • Defining increments may require definition of the complete system.

3.6 Spiral model

Advantages:	Disadvantages:
• The most flexible SDLC models in place. • Development phases can be determined by the project manager, according to the complexity of the project. • Project monitoring is very easy and effective. Each phase, as well as each loop, requires a review from concerned people. This makes the model more transparent. • High amount of risk analysis:Risk management is one of the in-built features of the model. • Changes can be introduced later in the life cycle as well. And coping with these changes isn't a very big headache for the project manager. • Project estimates in terms of schedule, cost etc become more and more realistic as the project moves forward and loops in spiral get completed.	• Cost involved in this model is usually high. • It is a complicated approach especially for projects with a clear SRS. • Skills required, to evaluate and review project from time to time, need expertise. • Rules and protocols should be followed properly to effectively implement this model. Doing so, through-out the span of project is tough. • Due to various customizations allowed from the client, using the same prototype in other projects, in future, is difficult. • It is not suitable for low risk projects. -:Meeting budgetary and scheduling requirements is tough if this

• It is suitable for high risk projects, where business needs may be unstable. • A highly customized product can be developed using this. • Good for large and mission-critical projects. • Software is produced early in the software life cycle.	development process is followed. • Amount of documentation required in intermediate stages makes management of project very complex affair.

3.7Concurrent Development Model:

Advantages:	Disadvantages:
• It's flexible – the number incremental releases can be determined by the project team. • Immediate feedback from testing. • New features can be added late in the project. • No surprises during formal validation because testing has been continuous .	• The SRS must be continually updated to reflect changes. • It requires discipline to avoid adding too many new features too late in the project.

References:

http://www.slideshare.net/DamianGordon1/software-engineering-methodologies

http://ptucse.loremate.com/se/node/2

*www.authorstream.com/.../aSGuest1156-96058-evolutionary-**model**s...*

itutils.hpage.com/get_file.php?id=1869954&vnr=804135

samples.jbpub.com/9780763785345/85345_CH04_Tsui.pdf

*www.cavehill.uwi.edu/staff/.../2009/**software_process_models**.htm*

*www.cs.rit.edu/~hpb/Scia/Bridge_2005/.../s2_**process_models**.pd*

*www.comp.dit.ie/rlawlor/.../ch03%20-%20**Software**%20**Process**es.ppt*

www.idi.ntnu.no/grupper/su/publ/pdf/capri-final.pdf

www.youtube.com/watch?v=YMbAdgb6pG8