Q.1what do you understand by the terms Software Process and Software Process Model?

Sol: **Software Process** is a series of identifiable stages that software product undergoes during its life time. Software process is sometimes refers as a software life cycle.

**Software Process Model** is the framework for defining the repeatable process that software engineering applies to the development of software.

While developing a software product, it is necessary for the development team to identify a suitable Process Model and adhere to it so that encourages development of software in a systematic and disciplined manner.

Q.2 Explain briefly each of the following software life cycle models with a suitable diagram.

2.1 Linear Sequential Model/Water fall Model

* The waterfall model derives its name due to the cascading effect from one phase to the other. In this model each phase well defined starting and ending point, with identifiable deliveries to the next phase.
* This model is sometimes referred to as **the linear sequential model** or **waterfall model.**
* The model consist of six distinct stages, namely:

1.      in the ***requirements analysis*** phase

(a)    The problem is specified along with the desired service objectives (goals)

(b)   The constraints are identified

  2.      In the ***specification*** phase the system specification is produced from the detailed definitions of (a) and (b) above. This document should clearly define the product function.

* Sometimes the requirements analysis and specifications phases are combined and represented as a single phase**.**

3.      In the system and software ***design*** *phase*, the system specifications are translated into a software representation. The software engineer at this stage is concerned with:

·        Data structure

·        Software architecture

·        Algorithmic detail and

·        Interface representations

* The hardware requirements are also determined at this stage along with a picture of the overall system architecture. By the end of this stage should the software engineer should be able to identify the relationship between the hardware, software and the associated interfaces. Any faults in the specification should ideally not be passed ‘down stream’

  4.      In the ***implementation and testing***phase stage the designs are translated into the software domain

·        Detailed documentation from the design phase can significantly reduce the coding effort.

·        Testing at this stage focuses on making sure that any errors are identified and that the software meets its required specification.

  5.      In the ***integration and system testing***phase all the program units are integrated and tested to ensure that the complete system meets the software requirements. After this stage the software is delivered to the customer

6.      The ***maintenance*** phase the usually the longest stage of the software. In this phase the software is updated to:

·        Meet the changing customer needs

·        Adapted to accommodate changes in the external environment

·        Correct errors and oversights previously undetected in the testing phases

·        Enhancing the efficiency of the software

**Advantages**

* Testing is inherent to every phase of the waterfall model
* It is an enforced disciplined approach
* It is documentation driven, that is, documentation is produced at every stage

**Disadvantages**

* The waterfall model is the oldest and the most widely used paradigm.  
  However, many projects rarely follow its sequential flow. This is due to the inherent problems associated with its rigid format. Namely:
* It only incorporates iteration indirectly, thus changes may cause considerable confusion as the project progresses.
* As The client usually only has a vague idea of exactly what is required from the software product, this WM has difficulty accommodating the natural uncertainty that exists at the beginning of the project.
* The customer only sees a working version of the product after it has been coded. This may result in disaster any undetected problems are precipitated to this stage.



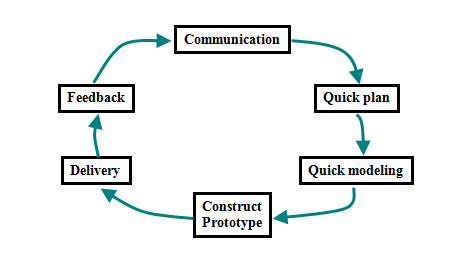
Fig: waterfall model

**2.2 Prototyping Model**

* 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.

**Need of Prototyping Model**

* This type of System Development Method is employed when it is very difficult to obtain exact requirements from the customer (unlike waterfall model, where requirements are clear). While making the model, user keeps giving feedbacks from time to time and based on it, a prototype is made. Completely built sample model is shown to user and based on his feedback; the SRS (System Requirements Specifications) document is prepared. After completion of this, a more accurate SRS is prepared, and now development work can start using Waterfall Model.



Fir: prototyping model.

**Advantages**

* When prototype is shown to the user, he gets a proper clarity and 'feel' of the functionality of the software and he can suggest changes and modifications.
* This type of approach of developing the software is used for non-IT-literate people. They usually are not good at specifying their requirements, nor can tell properly about what they expect from the software.
* It reduces risk of failure, as potential risks can be identified early and mitigation steps can be taken.
* Interaction between development team and client provides a very good and conductive environment during project.
* Time required to complete the project after getting final the SRS reduces, since the developer has a better idea about how he should approach the project.

**Disadvantages**

* Prototyping is usually done at the cost of the developer. So it should be done using minimal resources. It can be done using Rapid Application Development (RAD) tools.
* 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.

**2.3 Rapid Application Development Model**

* Rapid Application Development (RAD) is an incremental software development process model that emphasises a very short development cycle [typically 60-90 days].
* RAD is used primarily for information systems applications
* RAD approach includes the following phases:

## Business modelling

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

## Data modelling

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

## Process modelling

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

## Application generation

RAD assumes the use of the RAD fourth generation techniques and tools like VB, VC++, Delphi etc rather than creating software using conventional third generation programming languages. The RAD works to reuse existing program components (when possible) or create reusable components (when necessary). In all cases, automated tools are used to facilitate construction of the software.

## Testing and turnover

Since the RAD process emphasizes reuse, many of the program components have already been tested. This minimizes the testing and development time.

If a business application can be modularized so that each major function can be completed within the development cycle then it is a candidate for the RAD model. In this case, each team can be assigned a model, which is then integrated to form a whole.

## Disadvantages

* For Large (but scalable) projects, RAD requires sufficient resources to create the right number of RAD teams.
* If a system cannot be properly modularized, building components for RAD will be problematic
* RAD is not appropriate when technical risks are high, e.g. this occurs when a new application makes heavy use of new technology.

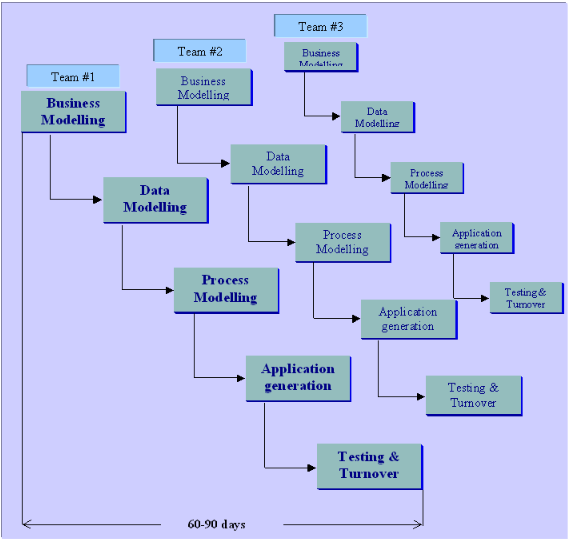


Fig: RAD model

**Advantages:**

* 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

**2.4 Evolutionary Software Process Models**

* The phases of the software construction are interleaved
* Feedback from the user is used throughout the entire process
* The software product is refined through many versions
* Develop an initial implementation, expose to users comments, refine until satisfied:
* Types of evolutionary development:
* **Exploratory development**
  + Start with requirements that are well defined
  + Add new features when customers propose new requirements
* **Throw-away prototyping**
  + Objective is to understand customer’s requirements (i.e. they often don’t know what they want, hence poor requirements to start
  + Use means such as prototyping to focus on poorly understood requirements, redefine requirements as you progress

**Advantages:**

* Happier customers since you help them define requirements
* Flexibility in modifying requirements
* Prototypes are very visual, hence no ambiguities

**Disadvantages:**

* Quick fixes may be involved
* “Invisible” process, not well-supported by documentation
* The system’s structure can be corrupted by continuous change

****

**2.5 Incremental Model:**

* Derives its name from the way in which the software is built. More specifically, the model is designed, implemented and tested as a series of incremental builds until the product is finished. A build consists of pieces of code from various modules that interact together to provide a specific function.
* This model combines the elements of the waterfall model with prototyping. However, unlike prototyping the IM focuses on the delivery of an operational product at the end of each increment.

**Note:**

* + Hybrid model that combines elements of the waterfall and evolutionary paradigms
  + The specification, design, and implementation phases are broken in smaller increments

**Advantages:**

* Provides better support for process iteration
* Reduces rework in the software construction process
* Allows early delivery of parts of the system
* Supports easier integration of sub-systems
* Lower risk of project failure
* Delivery priorities can be more easily set

**Disadvantages:**

* Increments need be relatively small
* Mapping requirements to increments may not be easy
* Common software facilities may be difficult to identify

**Applicability:**

* When it is possible to deliver the system “part-by-part”
* incremental approach is observed in the development of word processing applications where the following services are provided on subsequent builds:

1. Basic file management, editing and document production functions

2. Advanced editing and document production functions

3. Spell and grammar checking

4. Advance page layout



Fig: incremental model

**2.6 Spiral Model**

* The spiral model combines the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model, therein providing the potential for rapid development of incremental versions of the software.
* In this model the software is developed in a series of incremental releases with the early stages being either paper models or prototypes. Later iterations become increasingly more complete versions of the product.
* Depending on the model it may have 3-6 framework activities. Here ‘6-task region’ model is considered**.**

These regions are:

1. The **customer communication** task – to establish effective communication between developer and customer.
2. The **planning** task – to define resources, time lines and other project related information..
3. The **risk analysis** task – to assess both technical and management risks.
4. The **engineering** task – to build one or more representations of the application.
5. The **construction and release** task– to construct, test, install and provide user support (e.g., documentation and training).
6. The **customer evaluation** task – to obtain customer feedback based on the evaluation of the software representation created during the engineering stage and implemented during the install stage.

* An important distinction between the spiral model and other software models is the explicit consideration of risk. There are no fixed phases such as specification or design phases in the model and it encompasses other process models. For example, prototyping may be used in one spiral to resolve requirement uncertainties and hence reduce risks. This may then be followed by a conventional waterfall development.

**Advantages**

* The spiral model is a realistic approach to the development of large-scale software products because the software evolves as the process progresses.
* The developer and the client better understand and react to risks at each evolutionary level.
* The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.
* It maintains a systematic stepwise approach, like the classic life cycle model, but incorporates it into an iterative framework that more reflect the real world.
* If employed correctly, this model should reduce risks before they become problematic, as consideration of technical risks are considered at all stages.

**Disadvantages**

* Demands considerable risk.
* It has not been employed as much proven models (e.g. the WF model) and hence may prove difficult to ‘sell’ to the client.

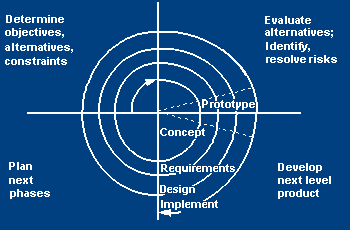


Fig: spiral model.

**2.7 Concurrent Development Model**

* Mention some advantages and disadvantages of each model.
* Similar to the Synchronize and Stabilize model used by Microsoft
* Multiple baselines are included as part of the process
* Ends with a formal validation phase with defined criteria for starting and stopping

**Advantages:**

* 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

**Disadvantages:**

* It requires discipline to avoid adding too many new features too late in the project

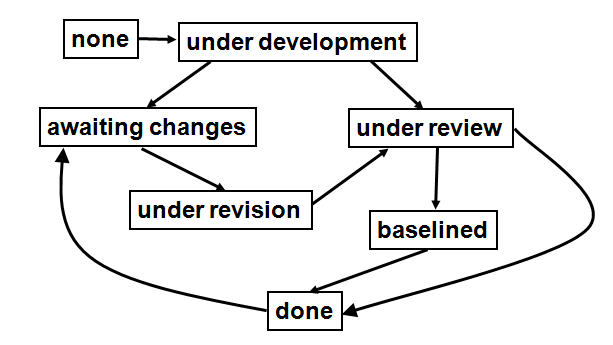


Fig: concurrent development model.

**References:**

* Software Process Models.htm
* Lecture notes
* PowerPoint slides from net.