**Information Systems Development Process (Software Development Life Cycle)**

**Phase 1 – Feasibility Study**

- Concerned with analyzing the benefits and solutions for the identified problem area
- Includes development of a business case, which determines the strategic benefits of implementing the system either in productivity gains or in future cost avoidance
- Identifies and quantifies the cost savings of the new system and estimates a payback schedule for the cost incurred in implementing the system or shows the projected return on investment (ROI)
- Within the feasibility study, the following typically are addressed:
  - Define a time frame for the implementation of the required solution.
  - Determine an optimum alternative risk-based solution for meeting business needs and general information resource requirements (e.g., whether to develop or acquire a system).
  - Determine if an existing system can correct the situation with slight or no modification (e.g., workaround).
  - Determine if a vendor product offers a solution to the problem.
  - Determine the approximate cost to develop the system to correct the situation.
  - Determine if the solution fits the business strategy.

**Phase 2 - Requirements Gathering**

- Concerned with identifying and specifying the business requirements of the system chosen for development during the feasibility study
- Requirements include descriptions of what a system should do, how users will interact with a system, conditions under which the system will operate, and the information criteria the system should meet
- This phase deals with the issues that are sometimes called nonfunctional requirements
- To accomplish the above in the requirements definition phase you should:
  - Identify and consult stakeholders to determine their expectations.
  - Analyze requirements to detect and correct conflicts and determine priorities.
  - Identify system bounds and how the system should interact with its environment.
  - Convert user requirements into system requirements (e.g., an interactive user interface prototype that demonstrates the screen look and feel).
  - Record requirements in a structured format.
  - Verify that requirements are complete, consistent, unambiguous, verifiable, modifiable, testable and traceable.
  - Resolve conflicts between stakeholders.
  - Resolve conflicts between the requirements set and the resources that are available.
Phase 3 – Software Design

- Key design phase activities include:
  - Developing system flowcharts and entity relationship models
  - Determining the use of structured design techniques
  - Describing inputs and outputs
  - Determining processing steps and computation rules
  - Determining data file or database system file design
  - Preparing program specifications
  - Developing test plans for the various levels of testing
  - Developing data conversion plans

Phase 4A – Development (only for bespoke software, not relevant for ERPs)

- Key activities performed in a development environment include:
  - Coding and developing program and system-level documents
  - Debugging and testing the programs developed
  - Perform unit testing
  - Developing programs to convert data from the old system for use on the new system
  - Creating user procedures to handle transition to the new system
  - Training selected users on the new system
  - Ensure modifications are documented and applied accurately

Phase 4B – Configuration (only relevant for ERPs not for bespoke software)

Configuring an ERP system is largely a matter of balancing the way the customer wants the systems to work. ERP systems build many changeable parameters that modify system operations. Configuration is to switch on and off features available in an ERP by means of parameterizing. Eg:- Inventory valuation could be calculated based on FIFO or LIFO method by the system based on the set configuration. Requirements that cannot be met by the standard ERP, will be treated as a customization by the vendor.

Phase 5 – Testing

Key activities performed in the testing stage are as follows.

- Create a test plan
- Create test scenarios and test cases
- Conduct integrated/system testing
- Report bugs
• Perform re-testing to ensure the closure of bugs identified after fixes are provided

Types of testing would include the following

✓ Alpha and beta testing
✓ Pilot testing
✓ White box testing
✓ Black box testing
✓ Function/validation testing
✓ Regression testing
✓ Parallel testing
✓ Sociability testing
✓ Automated testing

Phase 6 – Implementation

• During the implementation phase, the actual operation of the new information system is established and tested
• Final user-acceptance testing is conducted in this environment
• The system may also go through a certification and accreditation process to assess the effectiveness of the business application at mitigating risks to an appropriate level
• Develop support functions (1st and 2nd level support)
• End user training
• Carry out data migration – moving or copying data from the existing to the new system
• Changeover to the new system can be carried out based on parallel, abrupt or phased basis.

Phase 7 – Post Implementation Review

• A post-implementation review should meet the following objectives:
  ✓ Assess the adequacy of the system
  ✓ Evaluate the projected cost benefits or ROI
  ✓ Develop recommendations that address the system’s inadequacies and deficiencies
  ✓ Develop a plan for implementing the recommendations
  ✓ Assess the development project process

• A post-implementation review should be performed jointly by and independent party from those who implemented the new system, due to independency.

• A post implementation review should be performed after the system is stabilized and most issues are rectified (ideally 6 months to 1 year from the go-live date)

For all types of application development and ERP implementation, the above stated steps are applicable.
**Different life cycles used in software development**

1. **Waterfall life cycle**

   - Waterfall model describes a process of stepwise refinement (sequential)
     - Waterfall model takes a static view of requirements
       - Assumes complete requirements are clear at the beginning of project
     - Ignores changing needs.
     - Lack of user involvement once specification is written
     - Doesn't accommodate prototyping, reuse, etc.

2. **Prototyping life cycle**
• Prototyping is used for:
  – understanding the requirements with the involvement of the user
  – examining feasibility of a proposed design approach
  – Identify the objective of the project, obtain feedback and improve the system

• Problems:
  – users treat the prototype as the solution
  – a prototype is only a partial specification

3. Phased life cycle

• Incremental development
  – avoids ‘big bang’ implementation
  – but assumes all requirements known up-front

• Evolutionary development
  – allows for lessons from each version to be incorporated into the next
    • hard to plan for versions beyond the first;
    • lessons may be learnt too late
4. Spiral model

- The 4 main tasks of spiral models are
  - Determining objectives, alternatives and constraints
  - Evaluate alternatives, identify and resolve risks
  - Develop and verify next level product
  - Plan next phase
  - incorporates prototyping and risk analysis
  - Can show the working functional system before the system is completed
  - The life cycle can be repeated many times
  - The same activity may be repeated a number of time to clarify issues and obtain a precise requirements definition.

5. V-Model

- The V-model was originally developed from the waterfall software process model.
- The four main process phases – requirements, specification, design and implementation have a corresponding verification and validation testing phase.
- Implementation of modules is tested by unit testing, system design is tested by integration testing, system specifications are tested by system testing and finally acceptance testing verifies the requirements.
6. **Rapid Application Development (RAD)**

RAD is an incremental software development process model that emphasizes on extremely short development cycle (60-90 days). RAD is not appropriate to all the projects. It is more suitable for focused scope where the business objectives are well defined and narrow. Right skilled and attitude people are required with the appropriate tools to effectively achieve the output. Workshops (JAD Sessions) are held with all stakeholders under this methodology.

The RAD phases are as follows.
1. Business Modeling – The information flow between functions is defined such as who, where and what.
2. Data Modeling – The information collected from business functions is refined into a set of entities that are required for running the same.
3. Process Modeling – The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function.
4. Application Generation – Automated tools are used to facilitate construction of the software.
5. Testing and Turnover – Many of the programming components have already been tested since RAD emphasizes reuse.

7. **Agile Methodology**

Agile development refers to a family of similar development processes that adopt a nontraditional way of developing complex systems.

Agile development processes have a number of common characteristics, including:

a. The use of small, time-boxed subprojects or iterations
b. Replanning the project at the end of each iteration
c. Relatively greater reliance on tacit knowledge
d. Heavy influence on mechanisms to effectively disseminate tacit knowledge and promote teamwork
e. A change in the role of the project manager
During the annual maintenance of a project, when a change request is received from a user, the same shall be sent to the change control board, who will approve or reject the request, after doing an impact analysis (costing and effort for the request shall be analysed, against the return on investment). If the change request is approved, the software change shall be documented and the development carried out after which the normal SDLC cycle shall be followed.

Advantages and disadvantages of each life cycle method should be understood by the students.
Centralised versus Decentralised Information Systems

Centralised Information Systems - In centralized IS architecture, information resources are maintained on one or several large computers that are centrally controlled.

Advantages
1. High degree of control, Authority is at the top level. Data is maintained centrally.
2. Easy to maintain hardware, software, procedure, and operation standards.
3. Offers standardization and efficiency. Duplication of effort and resources is reduced. Saves cost and time due to efficiency.

Disadvantages
1. Bureaucracy and inflexible.
2. Slow operations

Decentralised Information Systems

In decentralized IS architecture, workers at different sites and departments (A, B, C) use information resources that are dedicated to their site or department.
Advantages
1. Allows departments and remote sites large degree of independence in organizing and utilizing their Information Systems (more flexible and effective).
2. Spreads authority to the lower organizational levels. Empowerment of individual business units.
3. Easy to tailor and change the system

Disadvantage
1. Difficult to share applications and data
2. Expensive to establish maintenance and service contracts with many vendors
3. Duplication of resources, effort and expertise
4. Poor decisions without senior management
5. Clashes and conflicting ideas may lead to delays and inefficiency

Distributed Information Systems
In distributed IS architecture, workers use the information resources of their own site or department, but can also use the resources of other sites or departments through communication lines. A distributed system in which it is partly centralized and partly decentralized.

Advantages
1. Each unit selects and implements its own system.
2. Remote units can share resources through communication lines.
3. Many organizations are changing to distributed architecture due to increased reliability and affordability of data communication and PC technology have encouraged this.
**Client/Server Architecture**

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations.

Computer transactions using the client/server model are very common. For example, to check your bank account from your computer, a client program in your computer forwards your request to a server program at the bank. That program may in turn forward the request to its own client program that sends a request to a database server at another computer to retrieve your account balance. The balance is returned back to the bank data client, which in turn serves it back to the client in your personal computer, which displays the information for the user.

The client/server model has become one of the central ideas of network computing. Most business applications being written today use the client/server model. So does the Internet's main program, TCP/IP. In marketing, the term has been used to distinguish distributed computing by smaller dispersed computers from the "monolithic" centralized computing of mainframe computers. However, this distinction has largely disappeared as mainframes and their applications have also turned to the client/server model and become part of network computing.

In the usual client/server model, one server, sometimes called a daemon, is activated and awaits client requests. Typically, multiple client programs share the services of a common server program. Both client programs and server programs are often part of a larger program or application. Relative to the Internet, your Web browser is a client program that requests services (the sending of Web pages or files) from a Web server (which technically is called a Hypertext Transport Protocol or HTTP server) in another computer somewhere on the Internet. Similarly, your computer with TCP/IP installed allows you to make client requests for files from File Transfer Protocol (FTP) servers in other computers on the Internet.

Other program relationship models included master/slave, with one program being in charge of all other programs, and peer-to-peer, with either of two programs able to initiate a transaction.

**Definition of client/server**

An architecture in which the user's PC (the client) is the requesting machine and the server is the supplying machine, both of which are connected via a local area network (LAN) or a wide area network (WAN) such as the Internet. Throughout the late 1980s and early 1990s, client/server was the hot buzzword as applications were migrated from minicomputers and mainframes with input/output terminals to networks of desktop computers.
With ubiquitous access to company LANs and the Internet, almost everyone works in a client/server environment today. However, to be true client/server, both client and server must share in the business processing. To understand this principle, follow the examples below of a query to a hypothetical database of a million records, each 1,000 bytes long. Notice the amount of data flowing over the network.

**Non-Client/Server**

In the above example, the database management system (DBMS) runs in the client, and the database is stored in the file server, which acts like a remote disk drive. Since no searching is done in the server, all one million records have to be sent over the network to the client for comparing. This is not "true" client/server because both sides are not sharing in the business processing.

**Not Client/Server**

In the above example, the database management system (DBMS) runs in the client, and the database is stored in the file server, which acts like a remote disk drive. Since no searching is done in the server, all one million records have to be sent over the network to the client for comparing. This is not "true" client/server because both sides are not sharing in the business processing.

**Two Tier Client/Server**

In the above example, the database management system (DBMS) runs in the client, and the database is stored in the server, which acts like a remote disk drive. Since no searching is done in the server, all one million records have to be sent over the network to the client for comparing. This is not "true" client/server because both sides are not sharing in the business processing.
Two-tier Client/Server

The above example is "true" client/server because both sides participate in the business processing. The database management system (DBMS) runs in the server. A query from the client is sent to the DBMS, which responds by searching the server and sending only results to the client. If 50 records matched the criteria in our million-record example, only 50 KB of data traverse the network rather than 1 GB.

Three-tier Client/Server

In the above example, processing is divided between two or more servers: one used for application processing and another for database processing. Because of the Internet, terms such as "Web based" and "Web enabled" replaced the 1990s client/server buzzword, and client/server implied old, legacy systems. However, although the client/server term may not be thrown around as much, Web-based systems today are entirely two-tier and three-tier client/server architectures. At the client side, the user’s PC executes scripts in Web pages. At the Internet side, Web servers and application servers process data before returning results to the user.