Module – 8

Software Reuse: Software reuse is the process of implementing or updating software systems using existing software assets.

- The systematic development of reusable components
- The systematic reuse of these components as building blocks to create new system

Advantages of reuse :

- Increase software productivity
- Shorten software development time
- Improve software system interoperability
- Develop software with fewer people
- Move personnel more easily from project to project
- Reduce software development and maintenance costs
- Produce more standardized software
- Produce better quality software and provide a powerful competitive advantage

Basic issues in reuse program :

The following are some of the basic issues that must be clearly understood for starting any reuse program.

- Component creation
- Component indexing and storing
- Component search
- Component understanding
- Component adaptation
- Repository maintenance

Reuse-based software engineering

- System reuse
  - Complete systems, which may include several application programs may be reused.

- Application reuse
  - An application may be reused either by incorporating it without change into other or by developing application families.
- Component reuse
  - Components of an application from sub-systems to single objects may be reused.
- Object and function reuse
  - Small-scale software components that implement a single well-defined object or function may be reused.

Software Reengineering: Software Re-engineering is a process of software development which is done to improve the maintainability of a software system. Re-engineering is the examination and alteration of a system to reconstitute it in a new form. This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing etc.

Objectives of Re-engineering:

- To describe a cost-effective option for system evolution.
- To describe the activities involved in the software maintenance process.
- To distinguish between software and data re-engineering and to explain the problems of data re-engineering.

Steps involved in Re-engineering:

1. Inventory Analysis
2. Document Reconstruction
3. Reverse Engineering
4. Code Reconstruction
5. Data Reconstruction
6. Forward Engineering

Client – Server Software Engineering:

The Client-server model is a distributed application structure that partitions task or workload between the providers of a resource or service, called servers, and

service requesters called clients. In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and deliver the data packets requested back to the client. Clients do not share any of their resources. Examples of Client-Server Model are Email, World Wide Web, etc.

Working process:

- **Client:** When we talk the word client, it mean to talk of a person or an organization using a particular service. Similarly in the digital world a client is a computer (Host) i.e. capable of receiving information or using a particular service from the service providers (servers).
- **Servers:** Similarly, when we talk the word servers, It mean a person or medium that serves something. Similarly in this digital world a server is a remote computer which provides information (data) or access to particular services.

Advantages of Client-Server software engineering:

- Centralized system with all data in a single place.
- Cost efficient requires less maintenance cost and Data recovery is possible.
- The capacity of the Client and Servers can be changed separately.

Disadvantages of Client-Server model:

- Clients are prone to viruses, Trojans and worms if present in the Server or uploaded into the Server.
- Server are prone to Denial of Service (DOS) attacks.
- Data packets may be spoofed or modified during transmission.
- Phishing or capturing login credentials or other useful information of the user are common and MITM(Man in the Middle) attacks are common.

Computer Aided Software Engineering: A CASE (Computer Aided Software Engineering) tool is a generic term used to denote any form of automated support for software engineering. In a more restrictive sense, a CASE tool means any tool used to automate some activity associated with software development. Many CASE tools are available. Some of these CASE tools assist in phase related tasks such as specification, structured analysis, design, coding, testing, etc.;

Reasons for using CASE tools:

- To increase productivity
- To help produce better quality software at lower cost

CASE environment:

CASE tools are characterized by the stage or stages of software development life cycle on which they focus. Since different tools covering different stages share common information, it is required that they integrate through some central repository to have a consistent view of information associated with the software development.

Benefits of CASE:

- A key benefit arising out of the use of a CASE environment is cost saving through all development phases. Different studies carry out to measure the impact of CASE put the effort reduction between 30% to 40%.

- Use of CASE tools leads to considerable improvements to quality. This is mainly due to the facts that one can effortlessly iterate through the different phases of software development and the chances of human error are considerably reduced.

- CASE tools help produce high quality and consistent documents. Since the important data relating to a software product are maintained in a central repository, redundancy in the stored data is reduced and therefore chances of inconsistent documentation is reduced to a great extent.

- CASE tools take out most of the drudgery in a software engineer's work. For example, they need not check meticulously the balancing of the DFDs but can do it effortlessly through the press of a button.