6.1 Software Project Concept

1. A **software project** is a planned effort to develop a software application within a specific time and budget.
2. It has a **defined goal**, such as developing a website, mobile app, or management system.
3. It includes **resources** like developers, tools, hardware, and money.
4. It follows a **structured process** (planning, development, testing, deployment).
5. It has **constraints** – time, cost, scope, and quality.
6. It involves **teamwork** among developers, testers, analysts, and managers.
7. The success of a project depends on **proper planning, communication, and risk management**.

6.2 Concept of Software Development Process

1. It is a **set of activities** used to develop software systematically.
2. It ensures software is built in a **planned and organized way**.
3. It includes stages like **requirement gathering, design, coding, testing, and maintenance**.
4. It helps in **reducing errors and improving quality**.
5. It defines **roles and responsibilities** of team members.
6. It improves **project control and monitoring**.
7. It ensures the final product meets **customer requirements**.

6.3 Concept of SDLC (Software Development Life Cycle)

SDLC (Software Development Life Cycle) is a structured (step by step) process used to develop high-quality software in a systematic and organized way. SDLC is a **framework** for developing high-quality software.

Phases of SDLC:

1. Planning

In this phase, the project idea is discussed and goals are defined.
Time, cost, and resources are estimated.
A basic project plan is prepared.

2. Feasibility Study

This phase checks whether the project is possible and practical.
It analyzes technical, financial, and operational feasibility.
It helps decide whether to continue the project or not. Types of Feasibility study.

- Technical feasibility – Is the required technology available?
- Economic feasibility – Is the project cost-effective?
- Operational feasibility – Will it work in the organization?
- Legal feasibility – Does it follow laws and regulations?

3. Requirement Analysis

- Detailed information is collected from users.
- Functional and non-functional requirements are identified.
- Requirements are documented clearly (SRS document).
- Misunderstandings are clarified with stakeholders.
- This phase defines what the system should do.

4. Design

- The system structure and architecture are planned.
- Database and user interface designs are prepared.
- It acts as a blueprint for developers.
-

5. Implementation (Coding)

- Developers write the actual program code.
- Modules are created based on the design.
- The system starts taking real shape.

6. Testing

- Testing is done to find and fix errors.
- Types of testing include unit testing, integration testing, and system testing.
- Ensures the software works correctly.

7. Deployment

- The software is installed in the user environment.
- It is made available for real use.
- Users may receive training if needed.
- It is delivered to the users.
- User training may be provided.
- The system becomes available for use.

8. Maintenance

- Errors found after deployment are fixed.
- System updates and improvements are made.
- New features may be added if required.
- Performance is monitored regularly.
- Ensures long-term working of the software.

6.4 System Analyst Vs Software Engineer

System Analyst

1. Studies and understands **business problems**.
2. Collects and analyzes **requirements from clients**.
3. Prepares **requirement specification documents**.
4. Acts as a **bridge between client and technical team**.
5. Focuses on **what the system should do**.
6. Suggests improvements in existing systems.
7. More involved in planning and communication.

Software Engineer

1. Designs and develops **software solutions**.
2. Writes and maintains **program code**.
3. Focuses on **how the system will be developed**.
4. Performs **testing and debugging**.
5. Uses programming languages and development tools.
6. Ensures system performance and reliability.
7. Responsible for technical implementation.

6.5 Requirement Collection Methods

1. **Interviews** – Face-to-face discussion with stakeholders.
2. **Questionnaires/Surveys** – Written questions for many users.
3. **Observation** – Watching users perform tasks.
4. **Document Review** – Studying existing system records.
5. **Workshops** – Group meetings for discussion.
6. **Brainstorming** – Generating ideas in teams.
7. **Prototyping** – Creating a sample system for feedback.

6.6 Concept of System Design

1. System design is the process of **planning how the system will work**.
2. It converts requirements into a **blueprint or structure**.
3. Includes **architectural design** (overall system structure).
4. Includes **database design** (data storage and relationships).
5. Includes **user interface design** (screen layouts and interaction).
6. Focuses on **security, performance, and scalability**.
7. Produces design documents to guide developers.

6.7 Software and Quality

1. Software quality means software works **correctly and efficiently**.
2. It must satisfy **user requirements**.
3. It should be **reliable and error-free**.

4. It should be **secure and protected from threats**.
5. It must be **user-friendly and easy to operate**.
6. It should be **maintainable and flexible**.
7. Quality is ensured through **testing, reviews, and standards**.

6.8 Software Development Models

1. Waterfall Model

1. Linear and sequential model.
2. Each phase is completed before the next begins.
3. Easy to understand and manage.
4. Suitable for small and clearly defined projects.
5. Difficult to make changes after development starts.
6. Heavy documentation is required.
7. Testing is done after coding phase.
8. The process does **not go back like streams of water**.
9. It flows only in one direction (downward).

---

2. Prototype Model

1. A sample (prototype) is created first.
2. It is mainly used to understand user requirements.
3. The prototype may not be the final product.
4. It helps clarify unclear requirements.
5. Focus is on requirement understanding.
6. A working model (prototype) is created first.
7. Helps users understand system functionality early.
8. User feedback is collected and improvements are made.
9. Reduces requirement misunderstandings.
10. Suitable when requirements are not clear.
11. Increases customer satisfaction.
12. Final system is developed after prototype approval.

---

3. Agile Model

1. Development is done in small parts called **iterations or sprints**.
2. Continuous interaction with customers.
3. Flexible and accepts changes easily.
4. Working software delivered frequently.
5. Emphasizes teamwork and collaboration.
6. Testing is done continuously.
7. Suitable for dynamic and large projects

---

4. Iterative Model

1. The complete system is developed in small parts (iterations).
2. Each iteration adds new features to the system.
3. Every version is a working product.
4. The system improves step by step until final completion.
5. Focus is on gradual development and improvement.