Module 1: Introduction to Software Processes & Quality

Overview

The first module lays the foundation for understanding **software development processes** and **software quality**. It introduces key software development life cycles (SDLC) and quality models that help ensure reliable and high-performing software.

1. Software Development Life Cycle (SDLC)

SDLC defines the structured approach to **planning, developing, testing, deploying, and maintaining software**. Different SDLC models exist, each suited to different project needs.

Common SDLC Models

Model	Description	Best Used For
Waterfall	Sequential process (Requirement → Design → Implementation → Testing → Deployment)	Well-defined, stable projects
V-Model (Verification & Validation)	Similar to Waterfall, but testing happens in parallel with development	Projects requiring strict validation
Agile	Iterative and incremental development (Scrum)	Fast-changing requirements, customer involvement
DevOps	Continuous development, integration, and deployment	Cloud-based, rapid release cycles
Spiral Model	Combines iterative development with risk assessment	High-risk, large-scale projects

Key Learning Points:

- SDLC models help teams **manage complexity, minimize risk, and improve efficiency**.
- Different projects require different SDLC models based on **complexity, flexibility, and risk management needs**.

2. Software Process Models

A software process model defines the framework for **developing and maintaining software**.

Key Software Process Models

1. **Plan-Driven (Traditional) Approach**
    - Follows a structured plan (e.g., Waterfall, V-Model).
    - Suitable for projects where requirements **don't change** often.
2. **Agile Process Models**
    - Encourages **continuous iteration, feedback, and collaboration**.
    - Used in **Scrum, Extreme Programming (XP)**.
3. **Hybrid Models**
    - Combines Agile with traditional methods for flexibility.
    - Example: **Agile-Waterfall Hybrid** (for projects with some fixed requirements and some evolving parts).

Key Learning Points:

- Choosing the right process model impacts **development speed, quality, and adaptability**.
- Agile and DevOps models are gaining popularity due to their **fast delivery cycles**.

3. Introduction to Software Quality Models

Software **quality** ensures that the final product meets user expectations and business requirements.

Common Software Quality Models

Model	Key Focus
ISO/IEC 25010	Defines software quality attributes like functionality, reliability, and security.

Model	Key Focus
McCall's Model	Focuses on product revision (maintainability), transition (portability), and operation (usability).
Boehm's Quality Model	Includes correctness, efficiency, and reliability.

ISO/IEC 25010: Key Software Quality Attributes

This model categorizes **quality into eight characteristics**:

1. **Functional Suitability** – Does the software do what it's supposed to?
2. **Performance Efficiency** – Does it run fast and use resources effectively?
3. **Compatibility** – Does it integrate well with other systems?
4. **Usability** – Is it user-friendly?
5. **Reliability** – Does it work consistently without failure?
6. **Security** – Is it protected against threats?
7. **Maintainability** – Is it easy to update and improve?
8. **Portability** – Can it be transferred to different environments easily?

Key Learning Points:

- **Software quality is not just about reducing defects**; it includes usability, performance, and maintainability.
- **Different quality models focus on different aspects**, such as user satisfaction, efficiency, or security.

4. Importance of Software Quality in Product Success

Poor software quality can lead to **financial loss, security breaches, and user dissatisfaction**.

Examples of Software Quality Failures

Case Study	Issue	Impact
Windows Vista (2007)	Poor performance and compatibility issues	Customers switched to other versions
Samsung Galaxy Note 7 (2016)	Battery explosion due to hardware & software issues	Massive recall, financial losses
Therac-25 (1985-87)	Software bug in radiation therapy machine	Overdoses of radiation, leading to deaths

Key Learning Points:

- **Poor software quality can cause product failure, legal issues, and reputational damage**.
- **Early detection of quality issues** through proper software processes can prevent major failures.

5. Case Study: Software Failures Due to Poor Processes

A real-world example helps students understand the **importance of structured processes and quality assurance**.

Case Study: Toyota Prius Software Recall (2014)

- **Issue:** Software bug caused sudden acceleration & braking problems.
- **Cause:** Poor **software validation & testing** before release.
- **Impact:** 1.9 million cars were recalled.
- **Lesson Learned:** Thorough **software testing & validation** is critical before deployment.

Key Learning Points:

- **Skipping quality assurance steps** can lead to **catastrophic failures**.
- **Software processes (like Agile, DevOps, and CI/CD)** help catch bugs earlier.