Software Engineering
2UCCE501

Module 4

Module 4

System Implementation, Configuration Management & Risk Management

4.1 Packages & Interfaces: Distinguishing between classes versus interfaces. Exposing class & package interfaces.

4.2 Mapping Model to code, Mapping object models to Database schema.

4.3 Component & Deployment Diagrams: Describing dependencies.

4.4 Managing & Controlling Changes: Managing & Controlling versions.

4.5 Categories of Risks. Nature of risks, Types of risks, Risk identification, Risk assessment, Risk Planning and control, Risk Management, Evaluating risk to schedule, PERT technique.

Packages & Interfaces

Packages:
A **package** is a way of grouping related classes, interfaces, and sub-packages together.

- Organize related classes and interfaces into a single unit.
- Declared using package keyword
- Does not support multiple inheritance.
- Group classes and interfaces based on functionality
- Syntax: package software.mypackage;

Packages help in:

- Modularity – breaking large projects into smaller logical units.
- Reusability – code in one package can be reused in another.
- Encapsulation – hiding implementation details inside a package.
- Namespace management – prevents name clashes between classes with the same name.

Packages & Interfaces

Interfaces:

- An interface is like a contract or blueprint.
- It defines what operations a class should perform but not how they are performed.
- A class that implements an interface must provide its own implementation for all methods in that interface.
- Interfaces support abstraction and polymorphism.
- Interface methods are implicitly public and abstract.
- Declared using interface keyword.
- Supports multiple inheritance as a class can implement multiple interfaces

Difference between class and interface

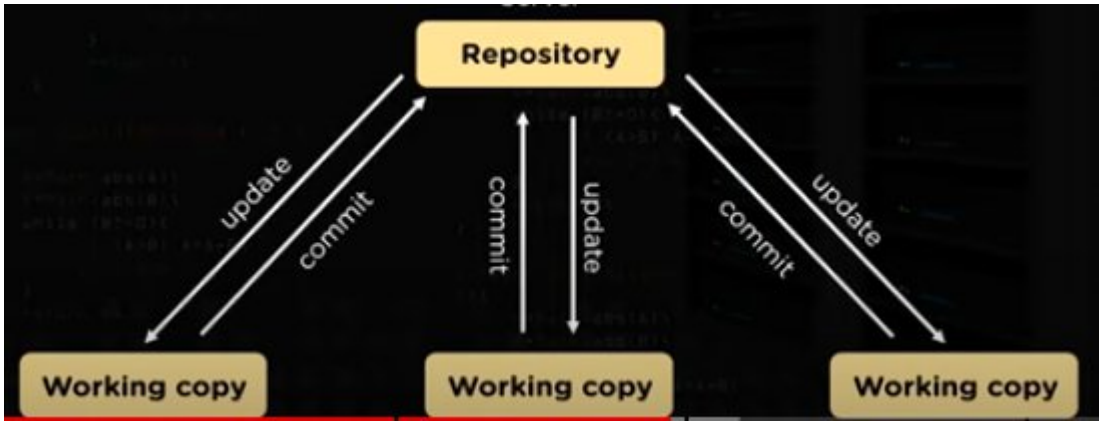
Aspect	Class	Interface
Definition	Blueprint to create objects with attributes (fields) and behavior (methods).	A contract that specifies only behaviors (methods), no state.
Implementation	Contains actual implementation (code inside methods).	No implementation (only method signatures).
Multiple Inheritance	A class can extend only one class (single inheritance).	A class can implement multiple interfaces.
Members	Can have fields (state) + methods (behavior).	Can only have constants + abstract methods (no instance fields).
Object creation	Objects can be created from a class.	Cannot create objects of an interface (only implemented by a class).

Version Control

- Challenges in distributed work environment:
  - Collaboration
  - Restoring previous version
  - What and where change happen
  - Backup
- A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made.

What is Version Control?

- A Version Control System records all the changes made to a file or set of files, so a specific version may be called later if needed



Version Control

- Version control **combines procedures and tools to manage different versions of configuration objects** that are created during the software process.

- A version control system implements four major capabilities:

(1) a **project database** that stores all relevant configuration objects

(2) a **version management capability** that **stores all versions** of a configuration object.

(3) a make facility that enables **construct a specific version of the software.**

(4) version control and change control systems often implement an **issues tracking (also called bug tracking)** capability

Version Control

- A number of version control systems **establish a change set**—a collection of all changes that are required to create a specific version of the software.

- **named change sets can be identified** for an application or system.

- construct a version of the software by specifying the change sets **(by name)**

Version Control

- Modeling approach for building new versions contains:

1. Template for building version
2. Construction rules
3. Verification rules

Change Control

- Change control is a systematic approach to manage all changes made to the product.
- It is used to identify, document, evaluate, approve (or reject), and implement changes in a project.
- It is part of Configuration Management.
- Too much change control and we create problems.
- large software project, uncontrolled change rapidly leads to chaos(confusion).
- For **large projects** change control combines **human procedures and automated tools** to provide a mechanism for the control of change.

Change Control

Engineering Change Order (ECO).

- An Engineering Change Order (sometimes called an Engineering Change Request, ECR) is the **formal document** that describes the details of a proposed change, its reason, and its impact.
- It usually comes after approval in the change control process.

ECO specifies:

- What needs to be changed (requirement, design, code, or document).
- Why the change is needed.
- Who approved it.
- Timeline and responsible engineers.

Change Control

Engineering Change Order (ECO).

Elements of change management:

1. **Access Control**

- Access control governs which software engineers have the authority to access and modify a particular configuration object.
- Prevents data breaches, misuse, and unauthorized changes.

Change Control

2. Synchronization Control.

- Mechanism to coordinate multiple users or processes trying to access a shared resource at the same time.
- Synchronization control helps to ensure that parallel changes, performed by two different people, don’t overwrite one another.
- Prevents conflicts, data corruption, or inconsistency.

3. Informal Change Control

- Developer makes changes in project.
- (Unofficial) Control that happens through communication, teamwork, and shared understanding, without strict procedures.

4. formal change control

- Control that is documented, structured, and follows official processes.
- Based on rules, standards, policies, and written procedures.
- is instituted when the software product is released to customers

Change Control

5. project level change control

- developer must gain approval from the project manager to makes changes.
- It is the process of managing, evaluating, and approving/rejecting changes that affect the scope, cost, schedule, or quality of a specific project.
- The goal is to ensure that only beneficial, necessary, and feasible changes are implemented, while avoiding uncontrolled “scope creep.”
- It’s about controlling project changes so that the project stays on track while adapting to valid new requirements.

Change Control Process in Software Configuration Management (SCM).