

SEN Most Repeated

Cohesion & Coupling

In software engineering, **Cohesion** and **Coupling** are key metrics for measuring modular design quality.

Cohesion refers to the degree to which the elements *inside a single module* belong together. High cohesion is good, meaning the module has a single, well-focused purpose.

Types of Cohesion (Best to Worst):

- **Functional:** Parts perform a single, well-defined task. (Excellent)
- **Sequential:** Output of one part is input to the next.
- **Communicational:** Parts operate on the same data.
- **Procedural:** Parts execute in a specific sequence.
- **Temporal:** Parts are related by timing (e.g., startup routine).
- **Logical:** Parts are logically similar but functionally different (e.g., a module handling all "input").
- **Coincidental:** Parts have no meaningful relationship. (Worst)

Coupling measures the interdependence *between modules*. Low coupling is good, meaning modules are independent and changes in one have minimal impact on others.

Types of Coupling (Worst to Best):

- **Content:** One module modifies another's internal data. (Worst)
- **Common:** Modules share global data.
- **External:** Modules share an externally imposed format.
- **Control:** One module controls the logic of another.
- **Stamp:** Modules share a composite data structure.
- **Data:** Modules communicate only through simple parameters. (Best)

Explain what is a risk? Different types of risk? and describe RMMM in detail?

1. Risk Definition

Risk refers to any uncertain future event that, if it occurs, will positively or negatively impact project objectives including scope, schedule, budget, and quality. It represents potential future problems or opportunities rather than current issues.

Key Characteristics:

- **Probability:** Likelihood of the risk occurring
- **Impact:** Magnitude of effect on project objectives

2. Types of Risk

Broad Categories:

- Business Risk: Operating cost coverage
- Non-Business Risk: Political/economic changes
- Financial Risk: Credit, liquidity, market risks

Project Risks:

- Project Risks: Threaten project plan
- Technical Risks: Threaten product quality
- Business Risks: Threaten product viability
- Known/Unknown Risks: Predictable vs unpredictable

3. RMMM Strategy

Risk Mitigation:

Proactive measures to reduce either the probability of risk occurrence or the impact if it occurs. This involves preventive actions taken before risks materialize.

- **Examples:** Cross-training team members, creating documentation, building prototypes, conducting training sessions, and implementing preventive controls.

Risk Monitoring:

Continuous process of tracking identified risks, watching for new risks, and

evaluating risk management effectiveness throughout the project lifecycle.

- **Methods:** Maintaining risk registers, conducting regular risk review meetings, monitoring risk triggers, and updating risk assessments.

Risk Management:

Reactive contingency planning that defines specific actions to be taken when risks materialize into actual issues, despite mitigation efforts.

- **Key Elements:** Detailed contingency plans, allocated contingency reserves (time and budget), clearly defined risk triggers, and assigned response responsibilities.
-

Change Control & Version Control

Change Control

Formal process for managing changes to project scope, requirements, or deliverables.

Key Elements:

- Change Request Submission
- Impact Analysis
- Approval/Rejection Decision
- Implementation Tracking
- Documentation Updates

Purpose: Prevents scope creep, maintains project stability, ensures stakeholder alignment.

Version Control

System for managing changes to files and code over time.

Key Features:

- Complete Change History
- Branching and Merging

- Conflict Resolution
- Rollback Capability
- Multi-user Collaboration

Benefits: Enables collaboration, maintains code integrity, tracks evolution.

Key Differences

Aspect	Change Control	Version Control
Focus	Project management	File/code management
Scope	Broad project changes	Specific file changes
Process	Formal approval required	Automated tracking
Tools	Manual systems/software	Git, SVN, Mercurial

Software Reengineering & Reverse Engineering

Software Reengineering

Process of improving existing software systems to enhance maintainability, reliability, and performance.

Objectives:

- Improve system structure
- Enhance documentation quality
- Update technology stack
- Increase maintainability
- Preserve business logic

Process Steps:

1. Inventory Analysis
2. Document Reconstruction
3. Reverse Engineering
4. Code Restructuring

5. Data Restructuring
6. Forward Engineering

Benefits:

- Cost-effective vs new development
- Preserves business rules
- Improves system quality
- Extends system lifespan

Reverse Engineering

Process of analyzing existing software to identify components and relationships.

Objectives:

- Understand system functionality
- Recover lost documentation
- Identify dependencies
- Extract design information
- Prepare for reengineering

Activities:

- Code Analysis
- Structure Recovery
- Documentation Generation
- Design Pattern Identification
- Interface Understanding

Key Differences

Aspect	Reengineering	Reverse Engineering
Purpose	Improve system	Understand system
Output	Enhanced software	Documentation/knowledge
Scope	Comprehensive process	Analysis phase only

Aspect	Reengineering	Reverse Engineering
Result	Working improved system	System understanding