1. **Test Coverage  
   Definition**: Test coverage is a measure of how much of the source code is exercised by a particular test suite.  
   **Importance**:

* Helps identify untested parts of a codebase.
* Encourages developers to write sufficient tests so that critical paths, branches, and features are all validated.
* Ensures better software quality, fewer regressions, and higher confidence in code changes.

1. **Change Request  
   Definition**: A change request is a formal proposal to modify any aspect of a product or system, often part of a change management process.  
   **Typical Uses**:

* Fixing defects or bugs.
* Adding new features or changing existing functionality.
* Adjusting requirements based on evolving user or business needs.

1. **Depth in Tree (DIT)  
   Definition**: In object‐oriented design, DIT (Depth of Inheritance Tree) is a metric that measures the levels of inheritance between the base class (root) and a given class.  
   **Usage**:

* A higher DIT can indicate more complexity in class behavior.
* A deeper inheritance chain might be harder to understand and maintain, but can also imply reuse of functionality.

1. **Failure Characteristics  
   Definition**: Failure characteristics describe how a system or component behaves when it fails.  
   **Examples**:

* **Failure modes** (how it fails—silent failure, crash, data corruption).
* **Failure impact** (the consequences of the failure—critical, partial, etc.).
* **Failure frequency** (how often or likely a failure occurs).
* These help in planning for resilience, fault tolerance, and disaster recovery.

1. **Mean Time Between Failures (MTBF)  
   Definition**: MTBF is a reliability metric indicating the average time between consecutive failures of a component or system.  
   **Usage**:

* Commonly used in hardware and systems engineering.
* Higher MTBF generally indicates higher reliability.
* Influences maintenance schedules and service level agreements (SLAs).

1. **Availability  
   Definition**: Availability is the proportion of time a system is operational and accessible.  
   **Formula**: Often calculated as:  
   Service Availability = MTBF / (MTBF + MTTR).  
   **Importance**:

* Critical in systems that must operate continuously (e.g., servers, networks).
* Tied to SLAs (e.g., 99.9% or “three‐nines” availability).

1. **Predictability  
   Definition**: Predictability in software processes or systems means the ability to estimate outcomes (such as time, cost, and quality) with reasonable accuracy.  
   **Relevance**:

* Helps with project planning, risk management, and resource allocation.
* Achieved through well‐established processes, metrics, and disciplined development practices.

1. **Project Measures  
   Definition**: Project measures (or project metrics) quantify various aspects of a project’s progress, performance, and quality.  
   **Common Examples**:

* **Schedule variance** (e.g., are we on time?).
* **Cost variance** (are we on budget?).
* **Defect metrics** (number of open bugs, time to fix).
* **Velocity** (in Agile methodologies).
* Helps ensure transparency, early detection of problems, and better planning.

1. **Software Design  
   Definition**: Software design is the process of defining the architecture, components, interfaces, and overall structure of a software system before coding.  
   **Importance**:

* A solid design leads to maintainable, extensible, and robust software.
* Encompasses high‐level architectural decisions and low‐level class, module, or API structures.

1. **Design Strategies  
   Definition**: Design strategies are approaches to solving software design problems. They can encompass architectural patterns, design patterns, and methodologies.  
   **Examples**:

* **Top‐down** vs. **bottom‐up** design.
* **Modular design**, **layered architecture**, **microservices**, etc.
* **GoF design patterns** (Singleton, Observer, Strategy, Factory, etc.).  
  **Key Consideration**: A good strategy balances complexity, performance, maintainability, and scalability.