Learning Resource
On
Software Project Management

Unit-3
Software Quality

Prepared By:
Kunal Anand
Assistant Professor, SCE
KIIT, DU, Bhubaneswar-24

Talkflow

- Importance of Software Quality
- Define Software Quality
- Product Quality vs Process Quality
- Quality Standards and Certifications
- Process and Issues in obtaining certifications
- Benefits and Implications of software quality for the organization and its customers
- Quality assurance vs Quality Control
- ISO 9126
- Process capability models (CMM; SPICE; Six Sigma; TMMi)
- Testing Quality plans
- Change management
- Challenges of Software Quality

Importance of Software Quality

- In software development, software quality is essential since it has a direct impact on operational effectiveness, customer satisfaction, and corporate success.
- High-quality software lowers the chance of malfunctions and expensive fixes by ensuring dependability, security, and maintainability.
- Software quality is crucial for the main reasons listed below:
  - Enhances User Satisfaction
  - Reduces Maintenance Costs
  - Improves Reliability and Performance
  - Ensures Security
  - Increases Business Revenue and Reputation
  - Compliance with Standards and Regulations
  - Facilitates Scalability and Future Growth

Software Quality

- **Software quality** refers to the degree to which a software product meets specified requirements, satisfies user needs, and performs reliably under expected conditions.
- It encompasses various attributes, including functionality, performance, security, maintainability, and usability.
- Key factors of software quality
  - **Correctness:** Must be functionally correct.
  - **Portability:** Should be easily portable to a different environment.
  - **Usability:** All users should be able to easily use the software.
  - **Reusability:** The components of the software should be reusable in future with some needed modifications.
  - **Maintainability:** A quality software product must be easy to maintain i.e., the error detection and correction could be done at ease.

Product Quality vs Process Quality

- Both **product quality** and **process quality** are essential in software development, but they focus on different aspects of quality management.
- **Product quality** focuses on the quality of the final software product, ensuring it meets user needs and requirements.
  - Its focus is on software functionality, performance, security, usability, and reliability.
  - Product quality aims to deliver a defect-free, efficient, and user-friendly product.
  - They can be evaluated through testing, user feedback, and performance metrics.
  - **Ex:** A mobile app that runs smoothly, meets functional requirements, and has no major bugs.

Product Quality vs Process Quality

- **Process quality** refers to the quality of the processes used to develop, test, and maintain the software. It focuses on the development methodologies, testing procedures, coding standards, and best practices.
  - They ensure a structured, efficient, and repeatable development process to minimize errors.
  - Process quality can be assessed using process maturity models like CMMI, ISO 9001, and Six Sigma.
  - **Ex:** Agile development, continuous integration, version control, and automated testing practices.
- A **high-quality process** usually leads to a **high-quality product** by reducing errors, improving efficiency, and ensuring consistency.
  - However, even a good process can produce a low-quality product if not properly followed or if requirements are poorly defined.

Quality Standards in Software Development

- **International Standards**
  - ISO/IEC 25010 (Software Product Quality Model)
  - ISO 9001 (Quality Management System – QMS)
  - ISO/IEC 27001 (Information Security Management System – ISMS)
  - ISO/IEC 12207 (Software Life Cycle Processes)

- **Capability Maturity Models**
  - CMMI (Capability Maturity Model Integration)
  - Six Sigma
  - TMMi (Test Maturity Model Integration)

Quality Certifications in Software Development

- **Individual Certifications**
  - Certified Software Quality Analyst (CSQA)
  - Certified Software Tester (CSTE)
  - ISTQB (International Software Testing Qualifications Board) Certifications
  - Certified Information Systems Security Professional (CISSP)
  - Project Management Professional (PMP)
- **Organizational Certifications**
  - ISO 9001 Certification
  - ISO/IEC 27001 Certification
  - CMMI Certification

Process of Obtaining Quality Certifications

- Obtaining a quality certification in software development requires organizations to follow a structured process while addressing challenges that arise during implementation.
- The certification process typically includes the following steps:
  - **Step 1: Identify the Relevant Certification** (Choose a certification based on business needs (e.g., ISO 9001 for quality management, CMMI for process maturity, ISO/IEC 27001 for security).
  - **Step 2: Conduct a Gap Analysis** (Compare current processes against certification requirements; Identify areas where improvements are needed to meet certification standards.)
  - **Step 3: Develop an Implementation Plan** (Define roles and responsibilities for compliance; Set milestones and deadlines for achieving quality improvements; Allocate resources (budget, personnel, training, and tools).

contd..

- **Step 4: Implement Process Improvements** (Establish or refine software development and quality assurance processes; Adopt best practices in project management, security, and documentation; Implement required tools for testing, monitoring, and compliance.)
- **Step 5: Employee Training and Awareness** (Educate employees on certification requirements and quality standards; Conduct workshops and training sessions to ensure compliance with new processes.)
- **Step 6: Internal Audits and Reviews** (Perform internal assessments to check compliance with certification standards; Identify non-conformities and take corrective actions before external evaluation.)
- **Step 7: Engage with a Certification Body** (Select an accredited certification body (e.g., ISO certification agencies, CMMI assessors); Submit required documentation and evidence of process compliance.)

contd..

- **Step 8: External Audit and Assessment** (Certification auditors evaluate the organization's adherence to quality standards; Organizations may need to demonstrate process controls, testing frameworks, and documentation; Auditors provide feedback, and organizations must address any non-compliance issues.)
- **Step 9: Certification Approval** (If all requirements are met, the certification body issues the certification; Certifications are usually valid for a certain period (e.g., ISO 9001 is valid for three years)).
- **Step 10: Continuous Improvement and Recertification** (Regularly monitor and improve processes to maintain certification; Conduct periodic internal audits and renew certification as required.)

Issues in Obtaining Quality Certifications

- **High Costs**
  - Certification involves costs related to assessments, training, documentation, and process improvements;
  - Small businesses may struggle with the financial burden of compliance.
- **Resistance to Change**
  - Employees may resist new quality processes due to unfamiliarity or increased workload;
  - Requires strong leadership and change management strategies.
- **Complexity of Compliance**
  - Standards like ISO 27001 (security) or CMMI Level 5 involve complex documentation and implementation;
  - Organizations may need external consultants for guidance.

Contd..

- **Time-Consuming Process**
  - The certification process can take months or even years, depending on the standard; Delays in implementation and audits may extend the timeline.
- **Lack of Skilled Personnel**
  - Many organizations lack internal expertise in quality management and compliance; Requires investment in employee training or hiring quality assurance specialists.
- **Meeting Industry-Specific Requirements**
  - Some industries (e.g., healthcare, finance) have additional regulatory requirements; Organizations may need multiple certifications (e.g., ISO 27001 + HIPAA for healthcare software).
- **Maintaining Certification**
  - Certification is not a one-time event; companies must maintain compliance through regular audits;

Benefits of software quality for the organization and its customers

- High-quality software has a significant impact on both organizations and customers. It enhances business performance, customer satisfaction, and long-term sustainability.
- Below are the key benefits and implications for both stakeholders.
  - **Benefits for Organizations:** Enhanced Reputation and Competitive Advantage; Reduced Costs and Higher Efficiency; Increased Revenue and Market Growth; Compliance with Industry Standards and Regulations; Better Employee Productivity and Morale
  - **Benefits for Customers:** Improved User Experience and Satisfaction; Increased Reliability and Security; Higher Productivity and Efficiency for Businesses; Reduced Costs for Customers; Long-Term Trust and Brand Loyalty

Benefits of software quality for the organization and its customers

- If an organization neglects software quality, the consequences can be severe for both the company and its customers.
  - **Implications for Organizations:** Increased costs due to frequent bug fixes, recalls, and legal actions; Negative customer reviews, loss of brand trust, and reduced market share; Higher vulnerability to cyberattacks and data breaches; Customers switch to competitors with better quality software; Non-compliance with standards may lead to legal fines and restrictions.
  - **Implications for Customers:** Waste of money on unreliable software that requires constant support; Loss of confidence in the product and company; Risk of data loss, identity theft, and financial fraud; Need to migrate to a different solution, causing inconvenience and extra costs; Users may face compliance issues if they rely on non-compliant software.

Quality Assurance vs Quality Control

- **Quality Assurance:** A proactive process that ensures software quality by improving development practices and preventing defects.
  - Ensures the development process follows best practices and standards.
  - Prevent defects from occurring.
  - Process audits, documentation reviews, training, standards enforcement, process improvement.
  - Everyone in the development team is responsible for QA.
  - Establishing coding standards; Conducting code reviews; Implementing Agile or DevOps best practices

contd..

- **Quality Control:** A reactive process that identifies and fixes defects in the software before release.
  - Ensures the final software meets quality requirements.
  - Detect and correct defects in the final product.
  - Testing (unit, integration, system, acceptance), defect identification, debugging.
  - Typically handled by the testing team or QC specialists.
  - Performing functional testing; Running security and performance tests; Bug tracking and reporting

contd..

- **Key Differences:**
  - QA is preventive (focused on processes), while QC is corrective (focused on the product).
  - QA is broader and applies throughout the Software Development Life Cycle (SDLC), whereas QC is specific to testing phases.
  - QA ensures processes lead to high-quality output, while QC verifies the final product meets expected standards.

ISO 9126 (Software Quality Model)

- ISO/IEC 9126 is an international standard that defines a framework for evaluating software quality.

- It was developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) to provide a structured approach to assessing software quality based on key characteristics.

- ISO 9126 has been replaced by ISO/IEC 25010, but it remains a foundational model for software quality assessment.

Contd..

- **Key Components of ISO/IEC 9126:** ISO 9126 defines four main parts:
  - **Software Quality Model** – Defines the characteristics and sub-characteristics of software quality.
  - **External Metrics** – Evaluates quality attributes as perceived by users.
  - **Internal Metrics** – Measures quality attributes using code analysis.
  - **Quality in Use Metrics** – Assesses software effectiveness in real-world scenarios.

Contd..

- Overview of ISO 9126 Quality Characteristics
  - **Functionality:** The ability of the software to meet user requirements and perform intended tasks.
    - **Sub-characteristics:** Suitability; Accuracy; Interoperability-; Security; Compliance
  - **Reliability:** The ability of the software to perform consistently under specified conditions.
    - **Sub-characteristics:** Maturity; Fault Tolerance; Recoverability
  - **Usability:** The ease with which users can interact with the software.
    - **Sub-characteristics:** Understandability; Learnability; Operability

Contd..

- **Efficiency:** The software's performance in relation to resource usage.
  - **Sub-characteristics:** Time Behavior (Response time, throughput); Resource Utilization

- **Maintainability:** The ease with which the software can be modified or improved.
  - **Sub-characteristics:** Analyzability; Changeability; Stability; Testability

- **Portability:** The ability of the software to function in different environments.
  - **Sub-characteristics:** Adaptability; Installability; Co-existence (Compatibility with other software); Replaceability

Contd..

- **Importance of ISO 9126**
  - Provides a standardized framework for software quality assessment.
  - Helps identify areas for improvement in software products.
  - Supports decision-making for developers, testers, and project managers.
  - Ensures software reliability, efficiency, and user satisfaction.

- Since ISO 9126 has been replaced by ISO/IEC 25010, organizations now follow the updated model, which expands on these concepts.

Process Capability Model

- In software quality, **process capability models** evaluate the maturity, efficiency, and predictability of software development and maintenance processes.
- These models help organizations improve software quality, reduce defects, and enhance productivity.
- Some of the widely used capability models are as follows:
  - Capability Maturity Model (CMM)
  - ISO/IEC 15504 (SPICE - Software Process Improvement and Capability Determination)
  - Six Sigma
  - Test Maturity Model Integration (TMMi)

SEI- Capability Maturity Model

- SEI Capability Maturity Model, SEI CMM, was proposed by software engineering institute of the Carnegie Melon University, USA.
- It was originally developed to assist the US dept. of defense in software acquisition.
- Gradually, many commercial organizations began to adopt CMM as a framework for their own internal improvement.
- CMM is a reference model for inducting the software process maturity into different levels.
- CMM is used in two ways: Capability evaluation and software process assessment.

SEI CMM Levels

- SEI CMM classifies the software companies into following maturity levels
  - **Level 1 (Initial)**
    - Very few or no defined and followed process
    - Engineers follow their own process which results in chaotic development efforts
    - Success of projects depend on individual effort and heroics. The successors face great difficulty in case if any individual leaves the company.
    - Low quality products
  - **Level 2 (Repeatable)**
    - Basic project management practices are established.
    - Necessary process discipline is in place to repeat the earlier success on projects with similar applications.

SEI CMM Levels

- **Level 3 (Defined)**
  - The process for both management and development activities are defined and documented.
  - Common understanding of activities, role, and responsibilities
  - **ISO 9000 aims at achieving this level**
- **Level 4 (Managed)**
  - Focus is on software metrics like product metrics and process metrics
  - Both process and product quality are measured to meet the specified quality requirements
- **Level 5 (optimizing)**
  - Process and product metrics are collected and analyzed for continuous process management
  - Lessons learnt from past projects are incorporated into the process
  - Continuous process improvement is the prime focus here.

Key Process Area (KPA) in CMM

- The **Key Process Areas,** KPA, of a level indicate the areas that an organization at the lower maturity level needs to focus to reach this level.
- KPAs provide a way for an organization to gradually improve its quality of over several stages.
- KPAs for each stage has been carefully designed such that one stage enhances the capability already built up.
  - Trying to focus on some higher level KPAs without achieving the lower level KPAs would be counterproductive.

Real-World Applications of CMMI

- Industries Using CMMI for Software Quality
  - **IT & Software Development:** Improves SDLC efficiency.
  - **Banking & Finance:** Ensures secure software and risk management.
  - **Healthcare & Pharma:** Reduces errors in medical software solutions.
  - **Government & Defense:** Ensures compliance and reliability in critical systems.

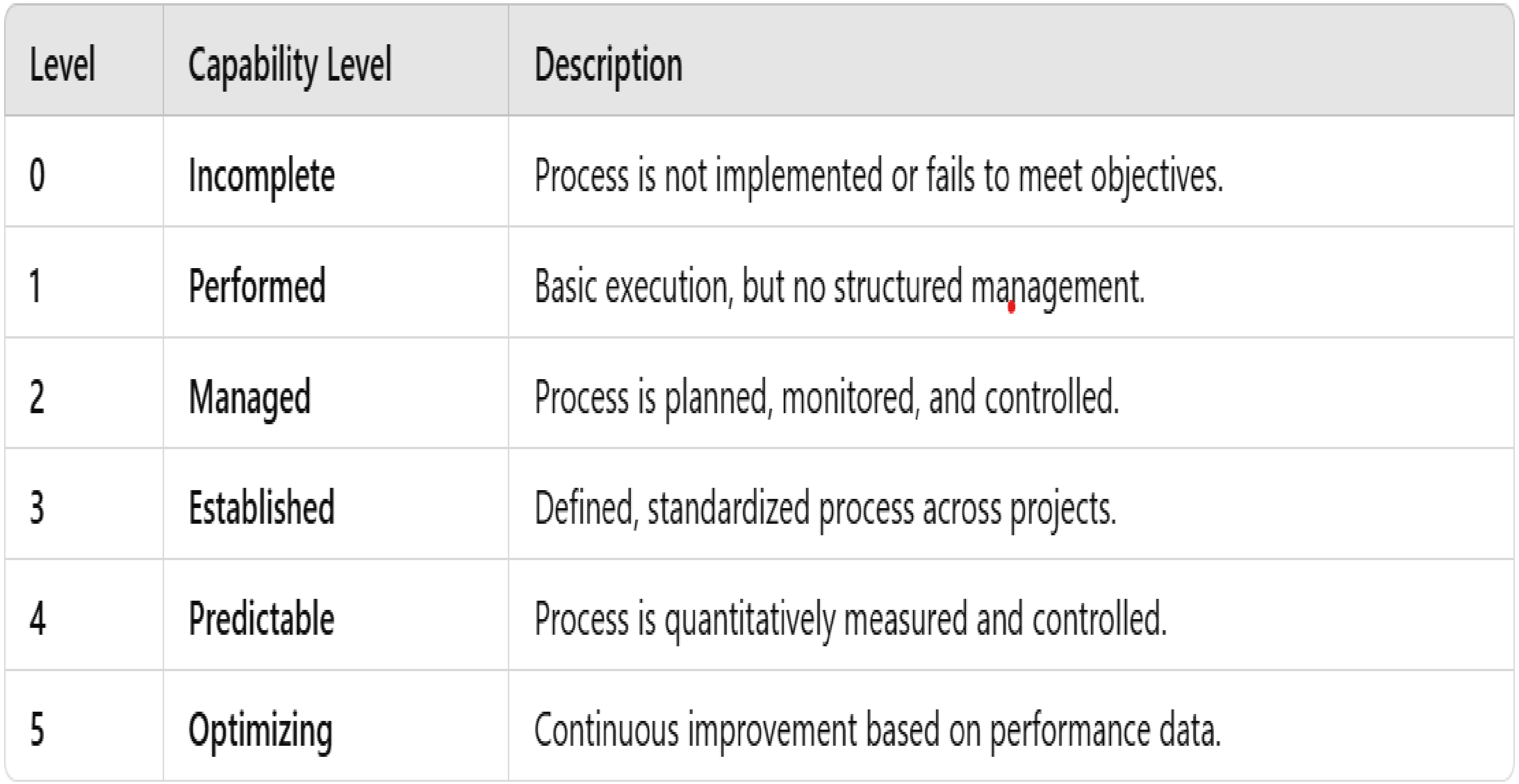
- **Example:** A software firm improved defect detection by 35% by moving from CMMI Level 2 to Level 4, implementing better test automation and risk assessment.

ISO/IEC 15504 (SPICE)

- ISO/IEC 15504, commonly known as SPICE **(Software Process Improvement and Capability Determination),** is an international standard for assessing and improving software development processes.
- It provides a structured approach to evaluating process maturity and capability in software engineering.
- **Structure of SPICE:** SPICE defines two key dimensions:
  - **Process Dimension:** SPICE categorizes software development into process areas, covering:
    - **Primary Lifecycle Processes:** Requirements, development, maintenance.
    - **Organizational Processes:** Process management, improvement.
    - **Supporting Processes:** Quality assurance, risk management.

Contd..

- **Capability Dimension:** Each process is evaluated on a six-level capability scale:



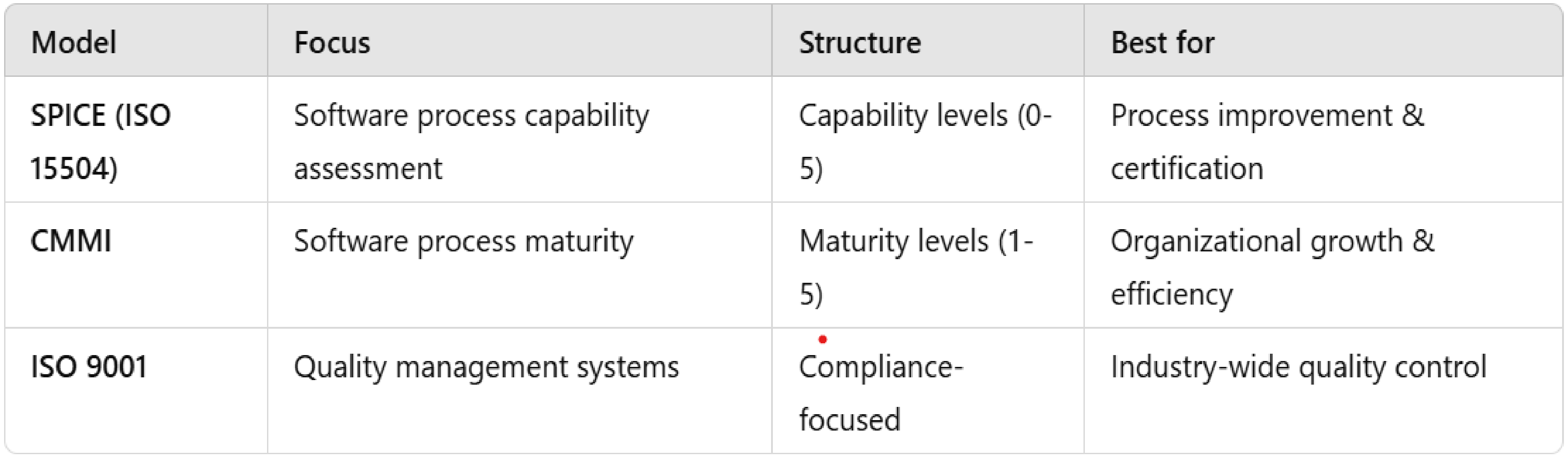
Level	Capability Level	Description
0	Incomplete	Process is not implemented or fails to meet objectives.
1	Performed	Basic execution, but no structured management.
2	Managed	Process is planned, monitored, and controlled.
3	Established	Defined, standardized process across projects.
4	Predictable	Process is quantitatively measured and controlled.
5	Optimizing	Continuous improvement based on performance data.

Key benefits of SPICE

- **Process Standardization:** Ensures consistency in software development.
- **Performance Improvement:** Helps identify and fix process weaknesses.
- **Risk Reduction:** Minimizes software defects and project failures.
- **Compliance & Certification:** Aligns with regulatory standards.
- **Competitive Advantage:** Enhances quality and efficiency in software delivery.

Contd..

SPICE vs Other Maturity Models



Model	Focus	Structure	Best for
SPICE (ISO 15504)	Software process capability assessment	Capability levels (0-5)	Process improvement & certification
CMMI	Software process maturity	Maturity levels (1-5)	Organizational growth & efficiency
ISO 9001	Quality management systems	Compliance-focused	Industry-wide quality control

Real-World Applications of SPICE

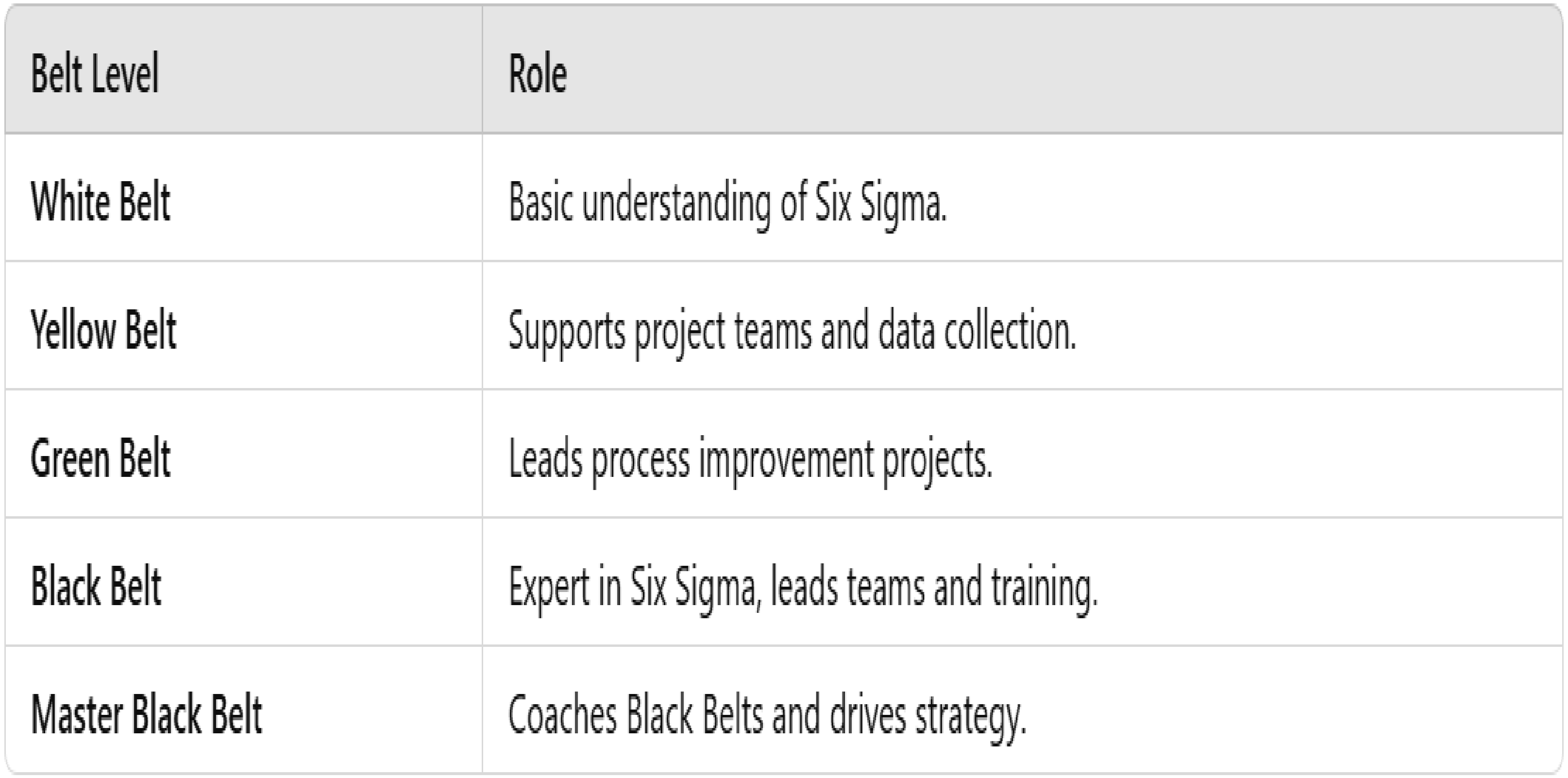
- **Automotive Software (ISO 26262 & ASPICE):** Ensures safety in car software.
- **Aerospace & Defense:** Compliance with stringent quality standards.
- **Banking & Finance:** Secure and reliable software development processes.
- **Healthcare IT:** Reduces risk in medical software solutions.

Six Sigma

- **Six Sigma** is a data-driven methodology that seeks to reduce variances and flaws in processes to improve them.
- Six Sigma was first created for manufacturing, but it is now frequently applied in IT and software development to improve customer happiness, quality, and efficiency.
- **Key Principles of Six Sigma:**
  - **Customer Focus:** Improve processes based on customer needs.
  - **Data-Driven Decision Making:** Use statistical methods to measure performance.
  - **Process Improvement:** Identify and eliminate defects.
  - **Eliminate Variability:** Reduce inconsistencies in software development.
  - **Continuous Improvement:** Aim for long-term process enhancement.

Six Sigma Levels (Belts)

- Six Sigma professionals are classified into different roles:



Belt Level	Role
White Belt	Basic understanding of Six Sigma.
Yellow Belt	Supports project teams and data collection.
Green Belt	Leads process improvement projects.
Black Belt	Expert in Six Sigma, leads teams and training.
Master Black Belt	Coaches Black Belts and drives strategy.

Six Sigma Methodologies

- **DMAIC (For Improving Existing Processes):** Used to enhance software development, testing, and maintenance.
  - **Define:** Identify software quality issues and goals.
  - **Measure:** Collect defect data, cycle time, and performance metrics.
  - **Analyze:** Find the root cause of software defects.
  - **Improve:** Implement solutions (e.g., better coding standards, test automation).
  - **Control:** Monitor improvements and prevent regression.

- **Use Case:** Reducing software defects, improving deployment efficiency.

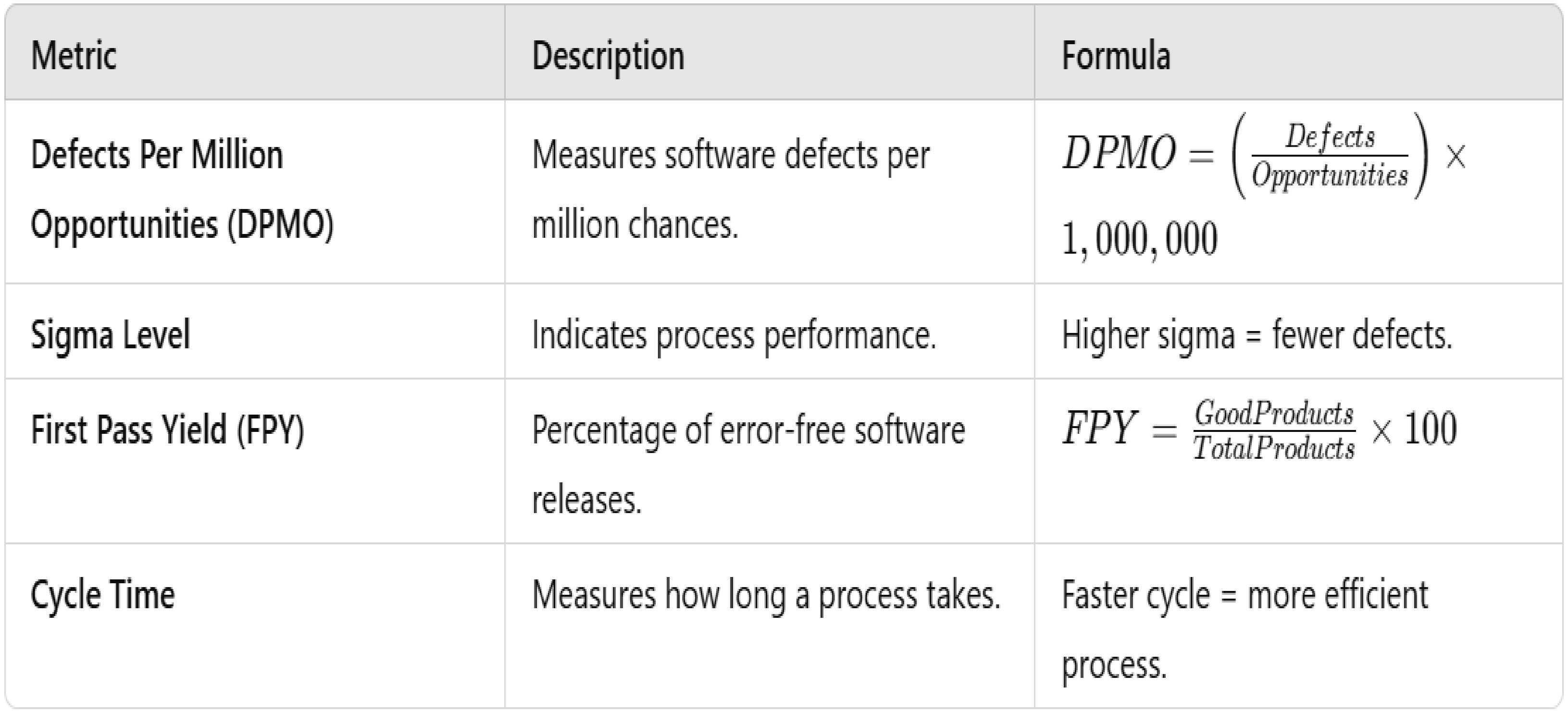
Contd..

- **DMADV (For Creating New Processes):** Used in new software projects or system designs.
  - **Define**: Set project goals and customer expectations.
  - **Measure**: Identify critical quality factors.
  - **Analyze**: Explore different software design alternatives.
  - **Design**: Develop an optimized software solution.
  - **Verify**: Test and validate the final product.

- Use Case: Designing a new application with minimal defects.

Six Sigma Metrics in Software Quality

- To measure Six Sigma effectiveness, these key metrics are used:

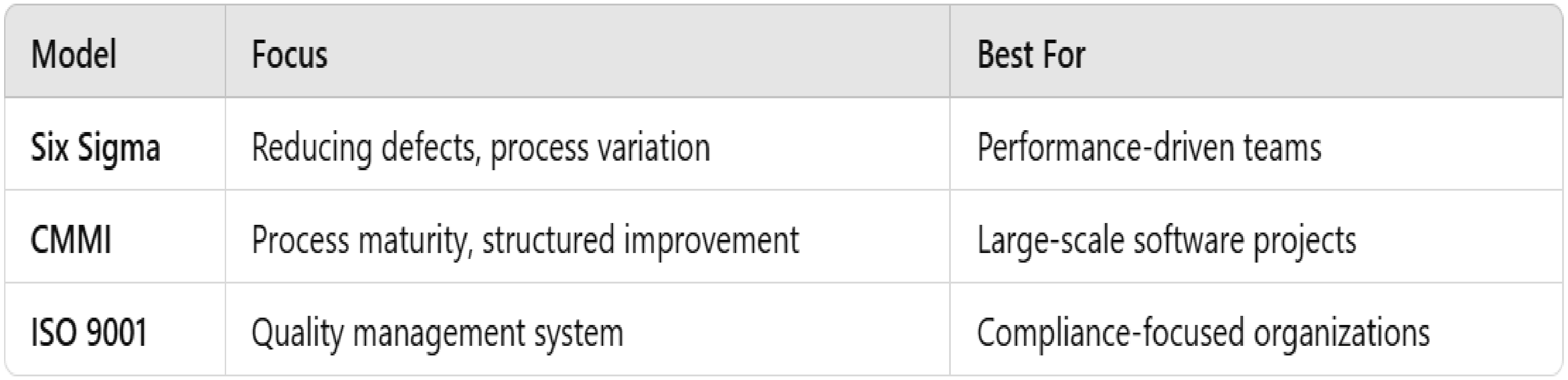


Metric	Description	Formula
Defects Per Million Opportunities (DPMO)	Measures software defects per million chances.	$DPMO = \left(\frac{Defects}{Opportunities}\right) \times 1,000,000$
Sigma Level	Indicates process performance.	Higher sigma = fewer defects.
First Pass Yield (FPY)	Percentage of error-free software releases.	$FPY = \frac{GoodProducts}{TotalProducts} \times 100$
Cycle Time	Measures how long a process takes.	Faster cycle = more efficient process.

- **Example:** A **Six Sigma Level 6** process has only 3.4 defects per million opportunities (DPMO).

Contd..

- **Six Sigma vs. Other Quality Models**



Model	Focus	Best For
Six Sigma	Reducing defects, process variation	Performance-driven teams
CMMI	Process maturity, structured improvement	Large-scale software projects
ISO 9001	Quality management system	Compliance-focused organizations

- **Real-World Applications of Six Sigma**

✔ Amazon – Uses Six Sigma to optimize software delivery and reduce outages.

✔ Microsoft – Improves software testing efficiency with defect reduction.

✔ IBM – Uses Six Sigma for IT process optimization.

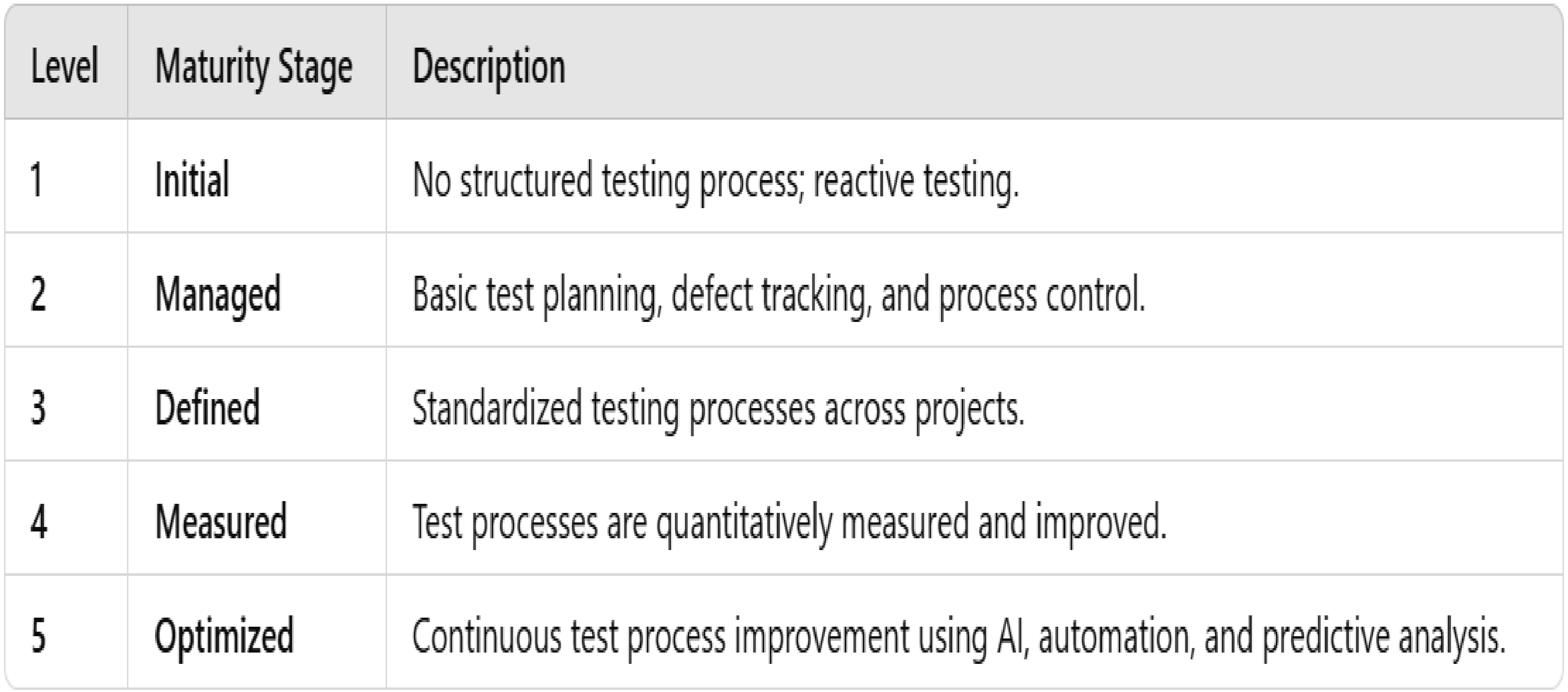
✔ Banking & Finance – Reduces transaction errors in digital banking systems.

Test Maturity Model Integration (TMMi)

- **TMMi**, developed by the TMMi Foundation, is a structured framework that provides best practices and process improvement strategies for software testing.
- It helps organizations evaluate their testing capability and move towards a mature, optimized testing process.
- Aligns closely with CMMI (Capability Maturity Model Integration).
- Focuses on test process improvement rather than just defect detection.

TMMi Maturity Levels

- TMMi defines **five levels** of test maturity, each representing a progression in testing capability.



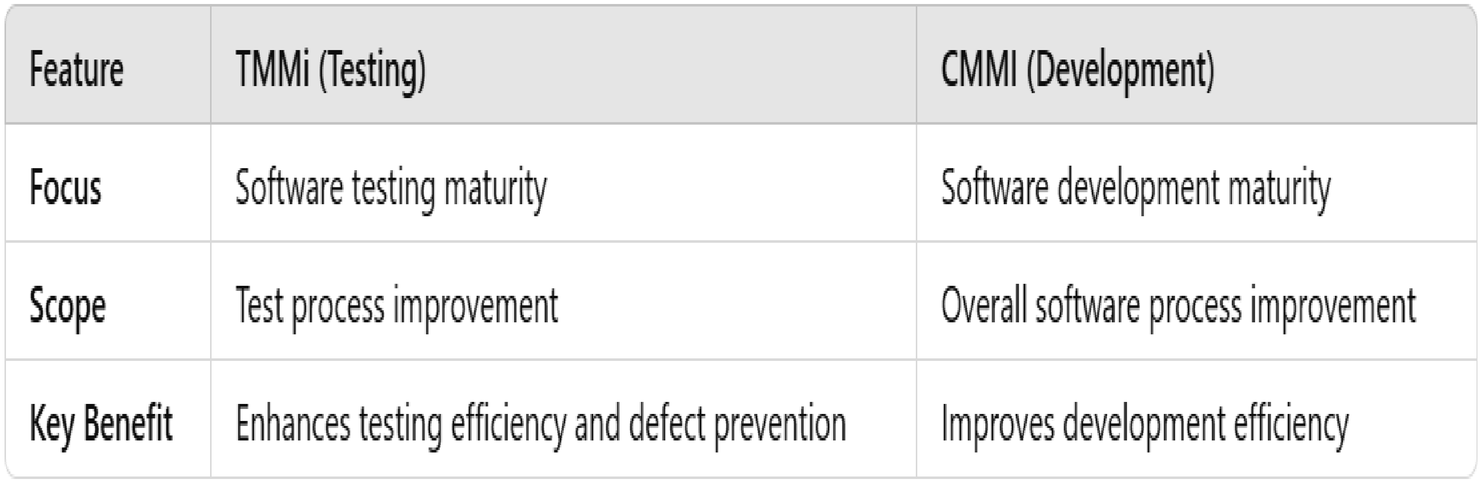
Level	Maturity Stage	Description
1	Initial	No structured testing process; reactive testing.
2	Managed	Basic test planning, defect tracking, and process control.
3	Defined	Standardized testing processes across projects.
4	Measured	Test processes are quantitatively measured and improved.
5	Optimized	Continuous test process improvement using AI, automation, and predictive analysis.

Key Areas of TMMi Implementation

- **Test Planning & Management**
  - Define structured test strategies and plans.
  - Improve test effort estimation and scheduling.
- **Test Process Standardization**
  - Establish consistent testing methods across projects.
  - Integrate testing with Agile, DevOps, and CI/CD pipelines.
- **Defect Prevention & Reduction**
  - Shift from reactive bug fixing to preventive testing.
  - Improve test case design and execution.
- **Test Automation & AI-driven Testing**
  - Implement automation frameworks for faster testing.
  - Use AI/ML for predictive defect analysis.

Contd..

- **How to Implement TMMi in an Organization?**
  - **Step 1:** Assess Current Test Maturity Level
  - **Step 2:** Define Improvement Goals
  - **Step 3:** Implement Best Practices
  - **Step 4:** Monitor and Measure Progress
  - **Step 5:** Achieve Continuous Test Process Improvement
- **TMMi vs. CMMI**



Feature	TMMi (Testing)	CMMI (Development)
Focus	Software testing maturity	Software development maturity
Scope	Test process improvement	Overall software process improvement
Key Benefit	Enhances testing efficiency and defect prevention	Improves development efficiency

Real-World Applications of TMMi

- **Real-World Applications of TMMi:** Industries Using TMMi for Software Testing

✔ Banking & Finance – Ensures secure, reliable transactions.

✔ Healthcare IT – Reduces defects in critical medical software.

✔ Automotive & Aerospace – Enhances embedded software reliability.

✔ Telecom & Retail – Improves mobile apps and e-commerce platforms.

- **Example:** A banking firm reduced defects by 40% by moving from TMMi Level 2 to Level 4, implementing automated regression testing and AI-based defect prediction.

Testing Quality Plans in Software Development

- A **Testing Quality Plan (TQP)** is a structured document that outlines the testing strategy, objectives, processes, and quality standards for a software project.
- It ensures that the final product meets quality expectations by defining test coverage, defect management, and validation processes.
- A TQP is designed to:
  - Ensure software quality through structured testing.
  - Define testing responsibilities and timelines.
  - Identify defects early, reducing costs and rework.
  - Align testing activities with business and project goals.
- **Example:** A well-defined Testing Quality Plan can reduce post-release defects by 40% through structured test case execution.

Key Components of a Testing Quality Plan

A comprehensive TQP typically includes the following sections:

- **Introduction & Scope:** Project Overview; Scope of Testing; Testing Objectives.
- **Testing Strategy & Approach:** Testing Levels; Testing Types; Test Environment
- **Test Planning & Execution:** Test Cases & Coverage; Entry & Exit Criteria
- **Defect Management & Reporting:** Defect Logging & Tracking; Severity & Priority Classification; Defect Resolution Process

Contd..

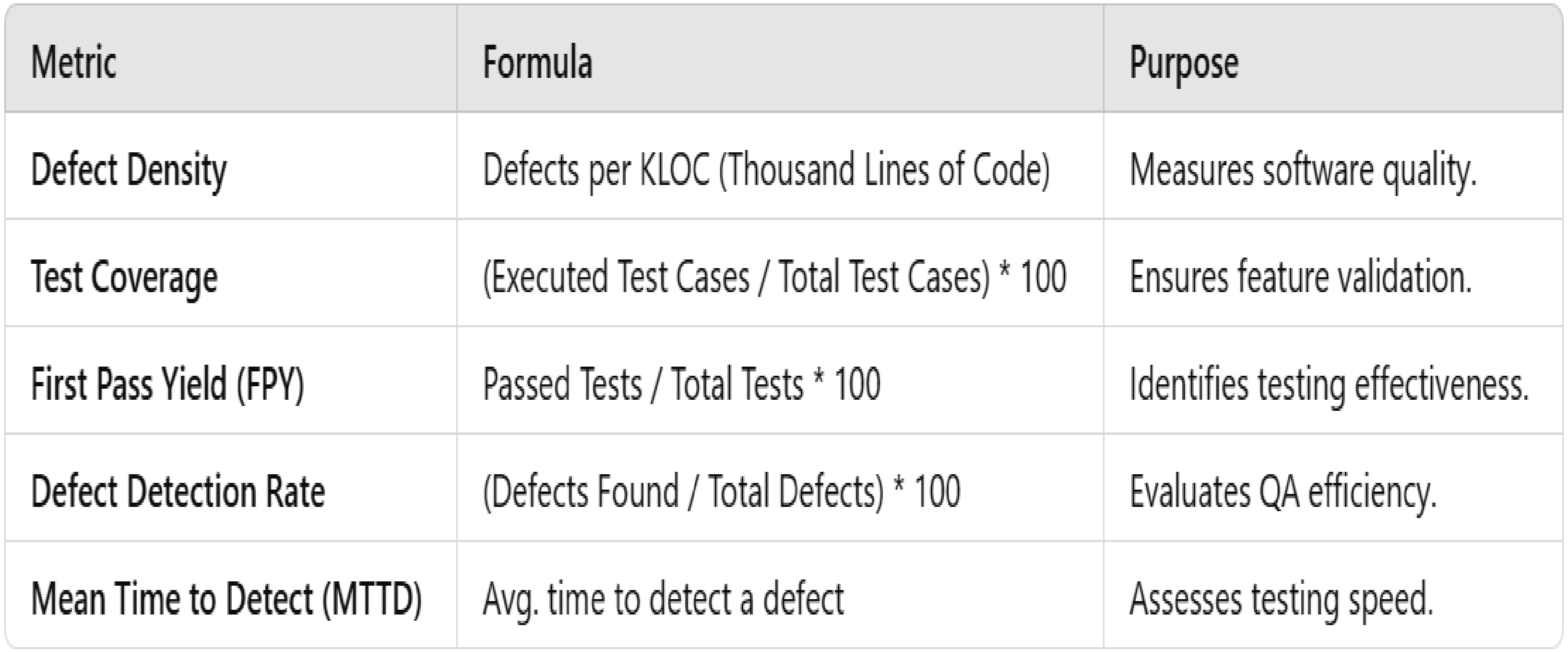
- **Roles & Responsibilities:**
  - Test Manager (Oversees test planning & execution)
  - Test Lead (Defines test strategy & ensures coverage)
  - QA engineers (Write & execute test cases, log defects)
  - Developers (Fix defects & assist in debugging)

- **Tools & Technologies:** Test automation (Selenium; Cypress); Performance testing (Jmeter; LoadRunner); Security testing (OWASP ZAP; Burp Suite)

- **Review & Approval Process:** Test Plan Reviews; Sign-Off Criteria

Test Quality Metrics

- To measure the effectiveness of the Testing Quality Plan, key metrics should be tracked:



Metric	Formula	Purpose
Defect Density	Defects per KLOC (Thousand Lines of Code)	Measures software quality.
Test Coverage	(Executed Test Cases / Total Test Cases) * 100	Ensures feature validation.
First Pass Yield (FPY)	Passed Tests / Total Tests * 100	Identifies testing effectiveness.
Defect Detection Rate	(Defects Found / Total Defects) * 100	Evaluates QA efficiency.
Mean Time to Detect (MTTD)	Avg. time to detect a defect	Assesses testing speed.

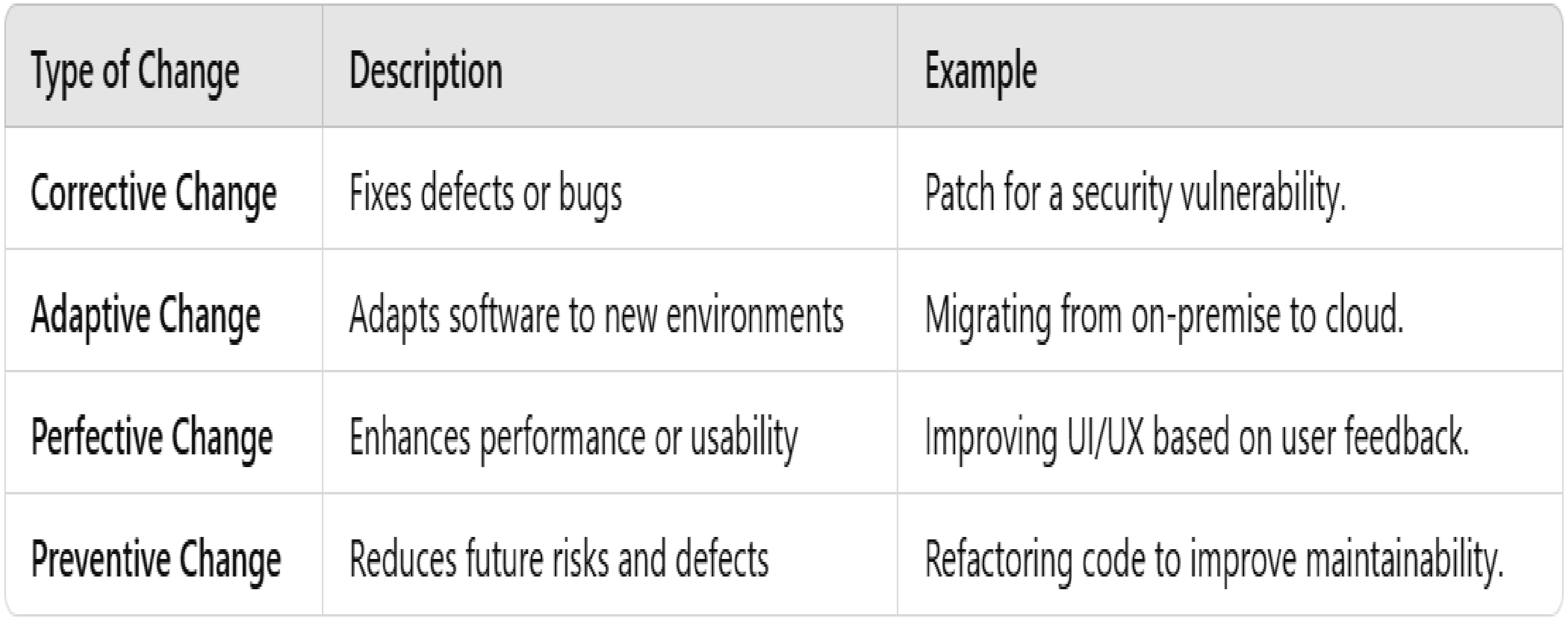
- **Example:** A higher Defect Detection Rate (≥85%) ensures early issue identification, reducing production failures.

Change Management

- **Change Management** in software quality ensures that modifications to software, processes, or systems are implemented smoothly, efficiently, and with minimal risk.
- It helps organizations maintain software reliability, security, and compliance while adapting to new requirements, technologies, and business needs.
- **Key Objectives:**
  - Ensure controlled, documented changes in software.
  - Minimize risks, defects, and downtime due to changes.
  - Maintain software stability and performance after updates.
  - Comply with regulatory and industry standards. (ISO, CMMI).

Types of Changes in Software Quality

- Changes in software quality can arise due to various internal and external factors.



Type of Change	Description	Example
Corrective Change	Fixes defects or bugs	Patch for a security vulnerability.
Adaptive Change	Adapts software to new environments	Migrating from on-premise to cloud.
Perfective Change	Enhances performance or usability	Improving UI/UX based on user feedback.
Preventive Change	Reduces future risks and defects	Refactoring code to improve maintainability.

- **Example:** A banking app undergoes adaptive changes to support new regulatory requirements, ensuring compliance with government policies.

Change Management Process

- A well-defined **change management process** ensures that every change is tested, documented, and implemented efficiently. Following are the steps to be followed for making any change:
  - **Change Request & Assessment** (Identify change and its impact on the software quality; Classify the change in terms of minor, major, or emergency)
  - **Change Approval & Planning** (Review by Change Control Board (Approve or Reject); If approved then define change implementation plan)
  - **Implementation & Testing** (Module will be modified and thoroughly tested by the member who requested for change)
  - **Post-Change Review & Documentation** (Change results will be reviewed by the CCB for approval or rejection; if approved then the baseline will be modified else the same baseline will continue)

Common challenges in Software Quality

- Here are some of the biggest challenges in maintaining software quality:
  - Incomplete or Changing Requirements
  - Poor Test Coverage & Inadequate Testing
  - Lack of Skilled QA & Development Teams
  - Security Vulnerabilities
  - Performance & Scalability Issues
  - Poor Defect Management & Tracking
  - Integration Challenges in Large-Scale Systems
  - Lack of Automation & Continuous Testing
  - Compliance & Regulatory Challenges