Quality Factors

McCall’s Quality Model (1977)

McCall’s Model was one of the first structured models for software quality. It was developed in 1977 by James A. McCall, Paul K. Richards, and Gene F. Walters under the U.S. Department of Defense.

It aimed to bridge the gap between users and developers by defining software quality in terms of three key perspectives:

1. **Product Revision** – How easy it is to modify the software. Focuses on software maintenance and changeability.
2. **Product Transition** – How adaptable the software is to new environments.
3. **Product Operation** – How well the software functions when used.

These perspectives contain **12 quality factors**, which define different aspects of software quality.

<table>
<tr><th>Perspective</th><th>Factor</th><th>Description</th><th>Example</th></tr>
<tr><td rowspan="5">Product Operation Factors (Ensures software works correctly)</td><td>Correctness</td><td>Does the software meet user requirements?</td><td rowspan="5">A banking application must be correct (calculate transactions accurately), reliable (run 24/7), efficient (use minimal CPU), secure (protect user data), and user-friendly.</td></tr>
<tr><td>Reliability</td><td>Can the software function without failure?</td></tr>
<tr><td>Efficiency</td><td>Does it use system resources optimally?</td></tr>
<tr><td>Integrity</td><td>How secure is the software from unauthorized access?</td></tr>
<tr><td>Usability</td><td>How easy is it to use?</td></tr>
<tr><td rowspan="3">Product Revision Factors (Ensures software is easy to maintain)</td><td>Maintainability</td><td>How easy is it to modify or debug?</td><td rowspan="3">A mobile app should allow easy bug fixes, adapt to new OS updates, and be simple to test.</td></tr>
<tr><td>Flexibility</td><td>Can the software adapt to new changes?</td></tr>
<tr><td>Testability</td><td>How easy is it to test and verify?</td></tr>
<tr><td rowspan="3">Product Transition Factors (Ensures adaptability to new environments)</td><td>Portability</td><td>Can it run on different platforms?</td><td rowspan="3">A cloud-based application should run on Windows, Linux, and macOS, allow code reuse, and integrate with third-party tools.</td></tr>
<tr><td>Reusability</td><td>Can parts of the software be reused in new projects?</td></tr>
<tr><td>Interoperability</td><td>Can it communicate with other systems?</td></tr>
</table>

ISO/IEC 25010

ISO/IEC 25010 is a software quality model developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). It replaces ISO/IEC 9126 and provides a comprehensive framework for evaluating software quality.

It defines:

1. **Product Quality:** How well the software meets functional and non-functional requirements.
2. **Quality in Use:** How well the software satisfies users in real-world scenarios.

ISO/IEC 25010 is the current global standard for software quality assessment.

1. Functional Suitability : Does the software meet functional requirements?
Sub-Characteristics:
- **Functional Completeness** – Does it provide all required features?
- **Functional Correctness** – Does it produce correct results?
- **Functional Appropriateness** – Is it suitable for its intended use?

Example: A banking system should accurately calculate interest, process transactions correctly, and provide all necessary services.

2. Performance Efficiency: Does the software use resources optimally?
Sub-Characteristics:
- **Time Behavior** – Does it have fast response time?
- **Resource Utilization** – Does it use CPU, RAM, and disk space efficiently?
- **Capacity** – Can it handle high loads without crashing?

Example: A stock trading application must execute orders in milliseconds and handle thousands of transactions per second.

3. Compatibility: Can the software work in different environments?
Sub-Characteristics:
- **Co-Existence** – Can it run alongside other software?
- **Interoperability** – Can it integrate with other systems?

Example: Google Docs integrates with Microsoft Word and Dropbox.

4. **Usability:** Is the software easy to use and learn?
Sub-Characteristics:
- **Appropriateness Recognizability** – Do users understand its purpose?
- **Learnability –** Can users quickly learn to use it?
- **Operability –** Is it easy to navigate?
- **User Error Protection –** Does it prevent mistakes?
- **User Interface Aesthetics –** Is the UI visually appealing?
- **Accessibility –** Is it usable by disabled individuals?

Example: iOS and Android UI focus on ease of use, clear icons, and intuitive navigation.

5. Reliability: Can the software run without failures?
Sub-Characteristics:
- **Maturity –** Is it stable and bug-free?
- **Availability –** Is it accessible 24/7?
- **Fault Tolerance –** Can it recover from failures?
- **Recoverability –** Can it restore data after a crash?

Example: Cloud services (e.g., AWS, Google Cloud) use redundant servers to ensure uptime.

6. **Security:** How protected is the software from threats?
Sub-Characteristics:
- **Confidentiality** – Does it protect sensitive data?

- **Integrity** – Does it prevent data corruption?
- **Non-repudiation –** Can actions be traced to users?
- **Accountability –** Does it log user activities?
- **Authenticity –** Does it verify user identity?

Example: Online banking apps use two-factor authentication and encryption.

7. Maintainability: How easy is it to modify and fix bugs?
Sub-Characteristics:

- **Modularity –** Can it be broken into smaller components?
- **Reusability –** Can code be used in other projects?
- **Analyzability –** Is it easy to debug?
- **Modifiability –** Can it be updated without breaking features?
- **Testability –** Is it easy to test?

Example: Microservices architecture allows modular development for easy maintenance.

8. Portability: Can the software run on different platforms?
Sub-Characteristics:

- **Adaptability –** Can it work without modifications?
- **Installability –** Is it easy to install?
- **Replaceability –** Can it replace other software without disruption?

Example: Web applications run on Windows, macOS, and Linux without changes.

Six Sigma

Six Sigma is a data-driven methodology designed to improve processes by reducing defects and variations. It was developed at Motorola in the 1980s and later adopted by General Electric (GE), Honeywell, and other Fortune 500 companies.
Purpose:

- Reduce process variation and defects.
- Improve efficiency, quality, and customer satisfaction.
- Achieve near perfection (3.4 defects per million opportunities – DPMO).

Six Sigma Methodologies
Six Sigma uses two key methodologies:

(A) DMAIC – For Process Improvement: Used for improving existing processes.

Phase	Purpose
D - Define	Identify the problem and project goals.
M - Measure	Collect data and analyze process performance.
A - Analyze	Identify root causes of defects.
I - Improve	Develop solutions and implement changes.
C - Control	Monitor the process to ensure long-term success.

Example: GE used DMAIC to reduce manufacturing defects by 50%.

(B) DMADV – For New Process/Design Creation: Used for designing new processes or products.

Phase	Purpose
D - Define	Identify customer requirements and project scope.
M - Measure	Gather data to define key performance indicators (KPIs).
A - Analyze	Develop and evaluate potential design alternatives.
D - Design	Select the best design and create a prototype.
V - Verify	Test, validate, and deploy the final design.

Example: Samsung used DMADV to design high-quality smartphones.

Six Sigma Levels (Belts)

Six Sigma certification follows a belt-based ranking system similar to martial arts.

Belt Level	Role & Responsibility
White Belt	Basic understanding of Six Sigma principles.
Yellow Belt	Supports projects, assists data collection.
Green Belt	Leads small improvement projects, works under Black Belts.
Black Belt	Leads complex Six Sigma projects, mentors Green Belts.
Master Black Belt	Trains and advises Black Belts, manages enterprise-wide initiatives.
Champion	Executive responsible for driving Six Sigma across the organization.

Example: Amazon’s Black Belt-certified employees lead process efficiency improvements in their warehouses