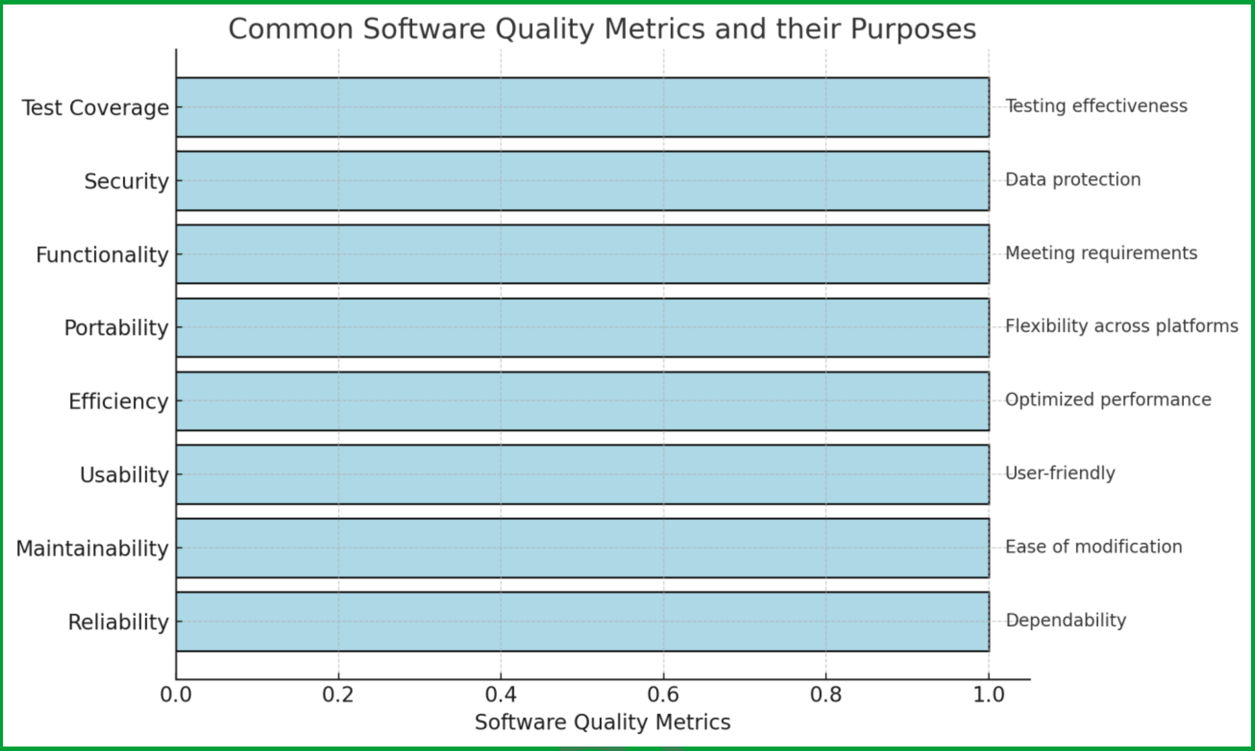
Metrics of Software Quality



Definition:

- Software quality metrics are measurements used to assess the quality of a software product or process.
- They help determine whether the software meets customer expectations and performs as intended.

Key Quality Metrics:

1. **Reliability:**
    - **Description:** Measures the ability of the software to perform its functions without failure under specific conditions.
    - **Metric Example:** Mean Time Between Failures (MTBF).
    - **Purpose:** Indicates the dependability of the software.
2. **Maintainability:**
    - **Description:** Assesses how easily software can be modified to correct issues, add new features, or improve performance.

- **Metric Example:** Mean Time to Repair (MTTR).
- **Purpose:** High maintainability reduces the cost and effort for updates and bug fixes.

3. **Usability:**
    - **Description:** Evaluates how easy it is for users to learn, operate, and interact with the software.
    - **Metric Example:** Time taken to learn, user error rates, user satisfaction scores.
    - **Purpose:** Ensures the software is user-friendly and meets end-user needs.
4. **Efficiency:**
    - **Description:** Measures the performance of the software in terms of resource usage (e.g., CPU, memory) and response time.
    - **Metric Example:** Response Time, Throughput.
    - **Purpose:** Ensures the software is optimized for performance.
5. **Portability:**
    - **Description:** Indicates the ability of software to run on different platforms or environments with minimal changes.
    - **Metric Example:** Number of platforms supported.
    - **Purpose:** Increases software flexibility and reach across multiple platforms.
6. **Functionality:**
    - **Description:** Evaluates whether the software provides all required functions correctly.
    - **Metric Example:** Percentage of requirements met.
    - **Purpose:** Ensures the software delivers all desired features and capabilities.
7. **Security:**
    - **Description:** Assesses the software's ability to protect against unauthorized access, breaches, and threats.
    - **Metric Example:** Number of vulnerabilities found, frequency of security patches.

- **Purpose:** Provides confidence that sensitive data and processes are protected.

8. **Test Coverage:**
   - **Description:** Measures how much of the software's code or functionality has been tested.
   - **Metric Example:** Percentage of code covered by tests.
   - **Purpose:** Indicates the effectiveness of testing processes.

Examples of Common Software Quality Metrics:

1. **Defect Density:**
   - **Description:** Number of defects found per unit of code (e.g., defects per thousand lines of code).
   - **Purpose:** Indicates overall software quality and helps identify areas that need improvement.
2. **Customer Satisfaction:**
   - **Description:** Measures the satisfaction of end-users or clients with the software.
   - **Metric Example:** Customer feedback scores, Net Promoter Score (NPS).
   - **Purpose:** Reflects how well the software meets user needs.
3. **Code Complexity:**
   - **Description:** Measures how complex the software's codebase is.
   - **Metric Example:** Cyclomatic Complexity (measures decision paths).
   - **Purpose:** Complex code is harder to maintain and more prone to errors.
4. **Code Coverage:**
   - **Description:** Indicates how much of the code is executed during testing.
   - **Purpose:** Higher coverage suggests more thorough testing and better reliability.

Benefits of Using Quality Metrics:

1. **Improved Quality Assurance:**
   - Metrics help identify weaknesses and areas that need improvement.

2. **Better Decision-Making:**
    - Helps managers make data-driven decisions on software enhancements.
3. **Customer Satisfaction:**
    - Ensures that the software meets user needs and quality expectations.
4. **Early Detection of Issues:**
    - Metrics can help detect problems early, reducing costs for late-stage fixes.

Challenges with Quality Metrics:

1. **Data Collection Effort:**
    - Collecting and analyzing metrics can be time-consuming and costly.
2. **Metric Interpretation:**
    - Misinterpretation of data can lead to incorrect conclusions.
3. **Balancing Metrics:**
    - Over-focusing on one metric may lead to neglecting other important aspects.