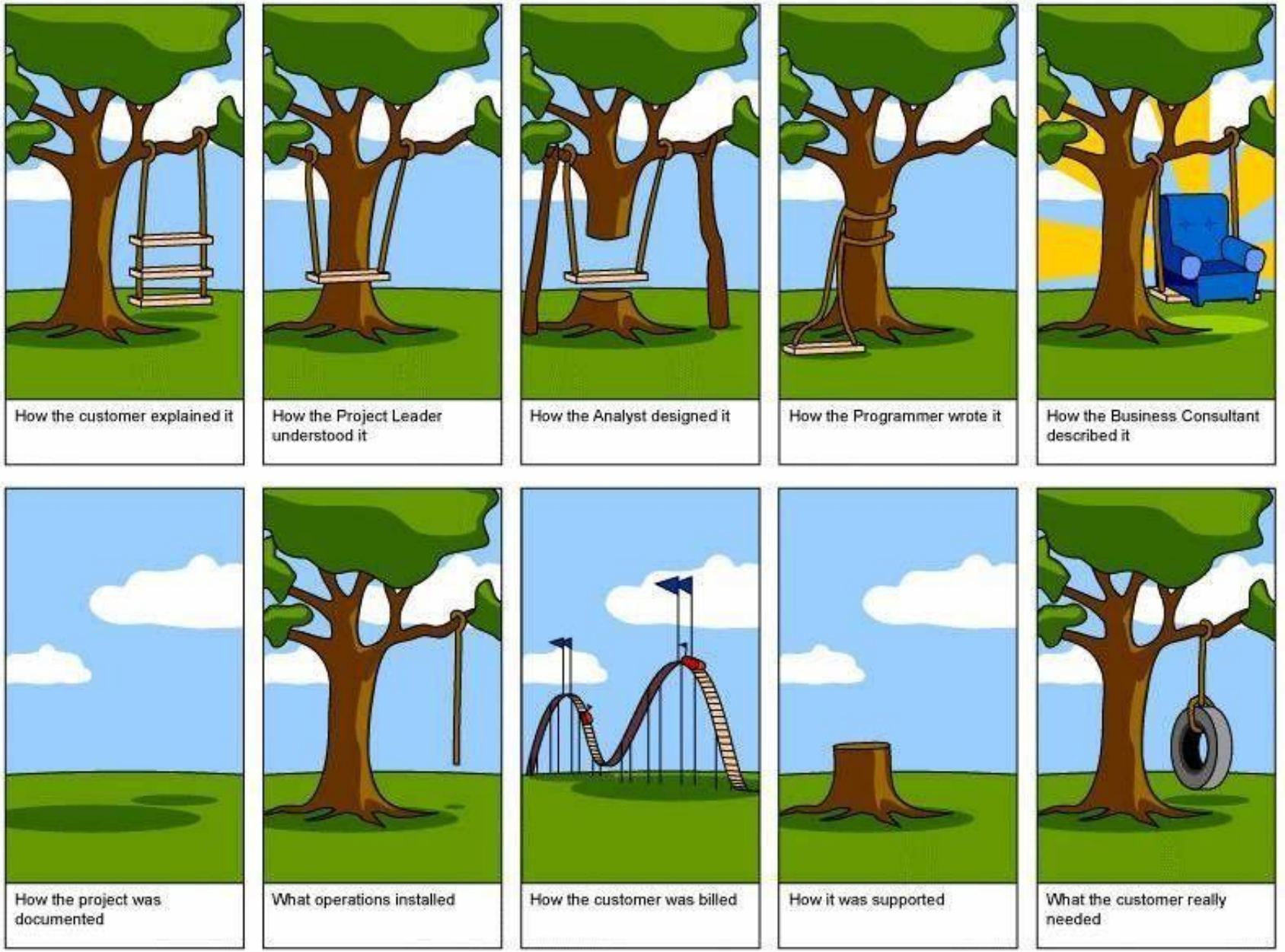
QUALITY ASSURANCE OF SOFTWARE

PMIT 6111: ST & QA


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

QUALITY ASSURANCE?

- **Software quality assurance (SQA)** is a process which assures that all software engineering processes, methods, activities and work items are monitored and comply against the defined standards. These defined standards could be one or a combination of any like ISO 9000, CMMI model, ISO15504, etc.
- SQA incorporates all software development processes starting from defining requirements to coding until release. Its prime goal is to ensure quality.

ELEMENTS OF SQA

- Standards
- Reviews and Audits
- Testing
- Error/defect collection and analysis
- Change management
- Education
- Vendor management
- Security management
- Safety
- Risk management

SQA GOALS

- **Requirements quality.** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- **Design quality.** Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality.** Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness.** A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

SQA GOALS, ATTRIBUTES AND METRICS

Goals	Attributes	Metric
Requirement quality	• Ambiguity • Completeness • Understandability • Volatility • Traceability • Model clarity	• Number of ambiguous modifiers (e.g., many, large, human-friendly) • Number of TBAs, TBDs • Number of sections/subsections • Number of changes per requirement • Time (by activity) when change is requested • Number of requirements not traceable to design/code • Number of UML models • Number of descriptive pages per model • Number of UML errors
Design quality	• Architectural integrity • Component completeness • Interface complexity • Patterns	• Existence of architectural model • Number of components that trace to architectural model • Complexity of procedural design • Layout appropriateness • Number of patterns used

SQA Goals, Attributes and Metrics

Goals	Attributes	Metric
Code quality	• Complexity • Maintainability • Understandability • Reusability • Documentation	• Cyclomatic complexity • Design factors • Percent internal comments • Variable naming conventions • Percent reused components • Readability index
QC effectiveness	• Resource allocation • Completion rate • Review effectiveness • Testing effectiveness	• Staff hour percentage per activity • Actual vs. budgeted completion time • Review metrics • Number of errors found and criticality • Effort required to correct an error • Origin of error

SQA PLAN

- Management section
  - describes the place of SQA in the structure of the organization
- Documentation section
  - describes each work product produced as part of the software process
- Standards, practices, and conventions section
  - lists all applicable standards/practices applied during the software process and any metrics to be collected as part of the software engineering work
- Reviews and audits section
  - provides an overview of the approach used in the reviews and audits to be conducted during the project
- Test section
  - references the test plan and procedure document and defines test record keeping requirements
- Problem reporting and corrective action section
  - defines procedures for reporting, tracking, and resolving errors or defects, identifies organizational responsibilities for these activities
- Other
  - tools, SQA methods, change control, record keeping, training, and risk management

THE CMMI PROCESS IMPROVEMENT FRAMEWORK

- The CMMI framework is the current stage of work on process assessment and improvement that started at the Software Engineering Institute in the 1980s.
- The SEI's mission is to promote software technology transfer particularly to US defence contractors.
- It has had a profound influence on process improvement
  - Capability Maturity Model introduced in the early 1990s.
  - Revised maturity framework (CMMI) introduced in 2001.

THE SEI CAPABILITY MATURITY MODEL

- Initial
  - Essentially uncontrolled
- Repeatable
  - Product management procedures defined and used
- Defined
  - Process management procedures and strategies defined and used
- Managed
  - Quality management strategies defined and used
- Optimising
  - Process improvement strategies defined and used

PROCESS CAPABILITY ASSESSMENT

- Intended as a means to assess the extent to which an organization's processes follow best practice.
- By providing a means for assessment, it is possible to identify areas of weakness for process improvement.
- There have been various process assessment and improvement models but the SEI work has been most influential.

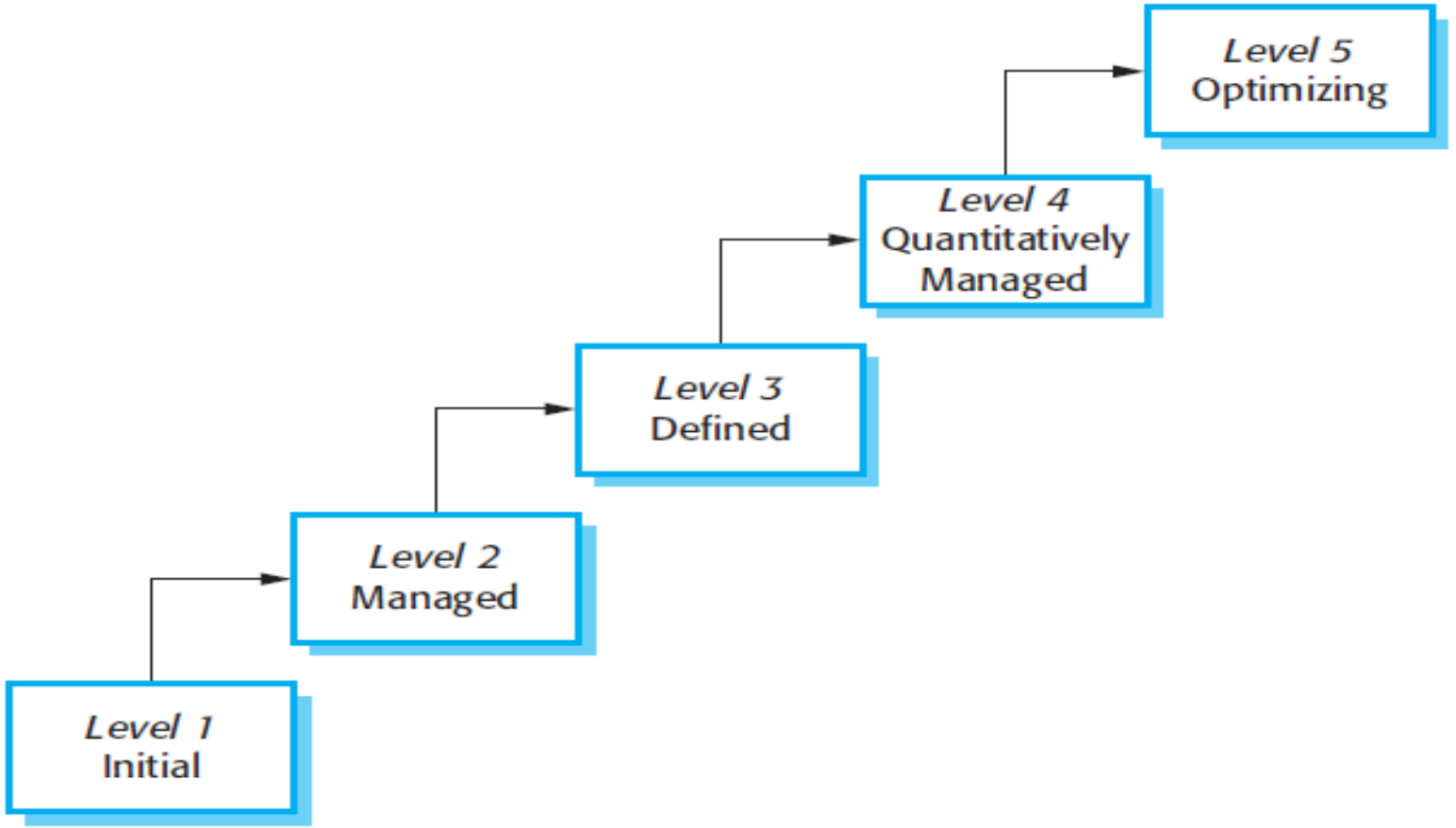
THE CMMI MODEL

- An integrated capability model that includes software and systems engineering capability assessment.
- The model has two instantiations
  - Staged where the model is expressed in terms of capability levels;
  - Continuous where a capability rating is computed.

CMMI MODEL COMPONENTS

- Process areas
  - 24 process areas that are relevant to process capability and improvement are identified. These are organized into 4 groups.
- Goals
  - Goals are descriptions of desirable organizational states. Each process area has associated goals.
- Practices
  - Practices are ways of achieving a goal - however, they are advisory and other approaches to achieve the goal may be used.

THE CMMI STAGED MATURITY MODEL



PROCESS AREAS IN THE CMMI

Category	Process area
Process management	Organizational process definition (OPD)
	Organizational process focus (OPF)
	Organizational training (OT)
	Organizational process performance (OPP)
	Organizational innovation and deployment (OID)
Project management	Project planning (PP)
	Project monitoring and control (PMC)
	Supplier agreement management (SAM)
	Integrated project management (IPM)
	Risk management (RSKM)
	Quantitative project management (QPM)

PROCESS AREAS IN THE CMMI

Category	Process area
Engineering	Requirements management (REQM)
	Requirements development (RD)
	Technical solution (TS)
	Product integration (PI)
	Verification (VER)
	Validation (VAL)
Support	Configuration management (CM)
	Process and product quality management (PPQA)
	Measurement and analysis (MA)
	Decision analysis and resolution (DAR)
	Causal analysis and resolution (CAR)

GOALS AND ASSOCIATED PRACTICES IN THE CMMI

Goal	Associated practices
The requirements are analyzed and validated, and a definition of the required functionality is developed.	Analyze derived requirements systematically to ensure that they are necessary and sufficient.
	Validate requirements to ensure that the resulting product will perform as intended in the user's environment, using multiple techniques as appropriate.
Root causes of defects and other problems are systematically determined.	Select the critical defects and other problems for analysis.
	Perform causal analysis of selected defects and other problems and propose actions to address them.
The process is institutionalized as a defined process.	Establish and maintain an organizational policy for planning and performing the requirements development process.

EXAMPLES OF GOALS IN THE CMMI

Goal	Process area
Corrective actions are managed to closure when the project's performance or results deviate significantly from the plan.	Project monitoring and control (specific goal)
Actual performance and progress of the project are monitored against the project plan.	Project monitoring and control (specific goal)
The requirements are analyzed and validated, and a definition of the required functionality is developed.	Requirements development (specific goal)
Root causes of defects and other problems are systematically determined.	Causal analysis and resolution (specific goal)
The process is institutionalized as a defined process.	Generic goal

CMMI ASSESSMENT

- Examines the processes used in an organization and assesses their maturity in each process area.
- Based on a 6-point scale:
  - Not performed;
  - Performed;
  - Managed;
  - Defined;
  - Quantitatively managed;
  - Optimizing.

THE STAGED CMMI MODEL

- Comparable with the software CMM.
- Each maturity level has process areas and goals. For example, the process area associated with the managed level include:
  - Requirements management;
  - Project planning;
  - Project monitoring and control;
  - Supplier agreement management;
  - Measurement and analysis;
  - Process and product quality assurance.

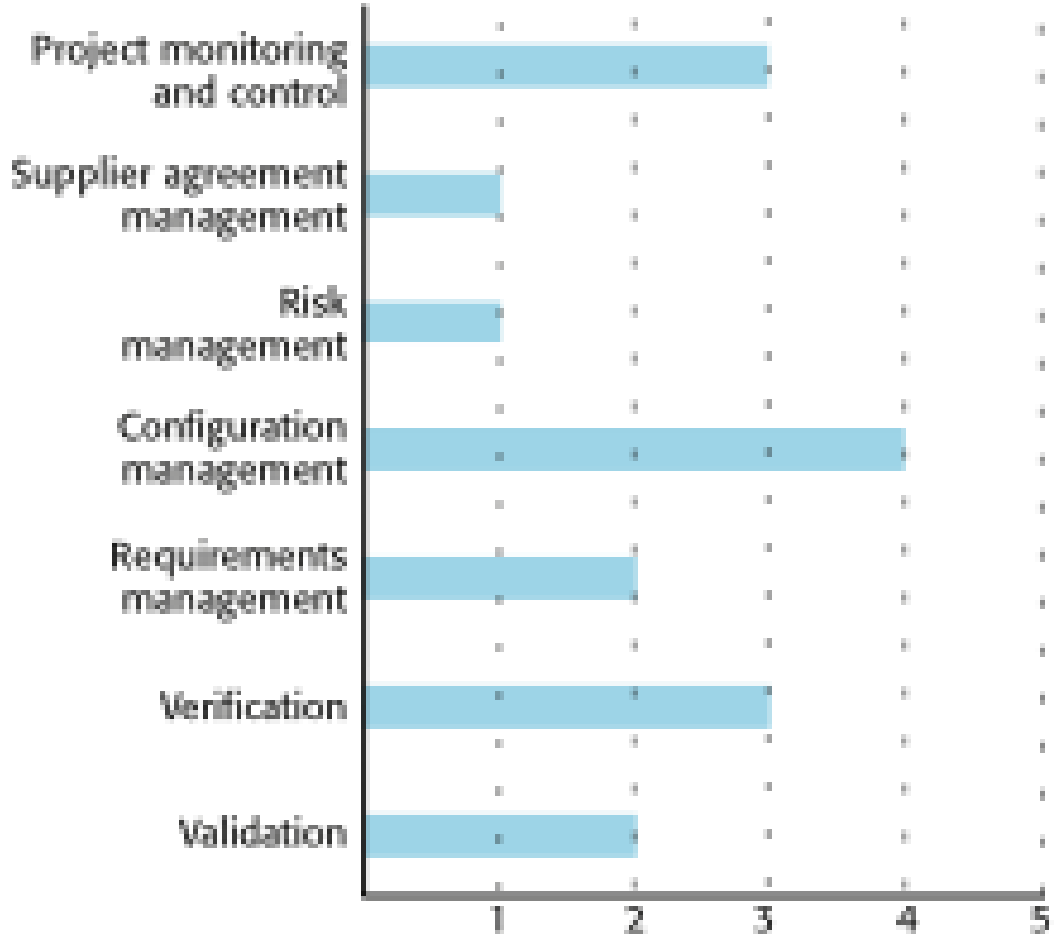
INSTITUTIONAL PRACTICES

- Institutions operating at the managed level should have institutionalized practices that are geared to standardization.
  - Establish and maintain policy for performing the project management process;
  - Provide adequate resources for performing the project management process;
  - Monitor and control the project planning process;
  - Review the activities, status and results of the project planning process.

THE CONTINUOUS CMMI MODEL

- This is a finer-grain model that considers individual or groups of practices and assesses their use.
- The maturity assessment is not a single value but is a set of values showing the organizations maturity in each area.
- The CMMI rates each process area from levels 1 to 5.
- The advantage of a continuous approach is that organizations can pick and choose process areas to improve according to their local needs.

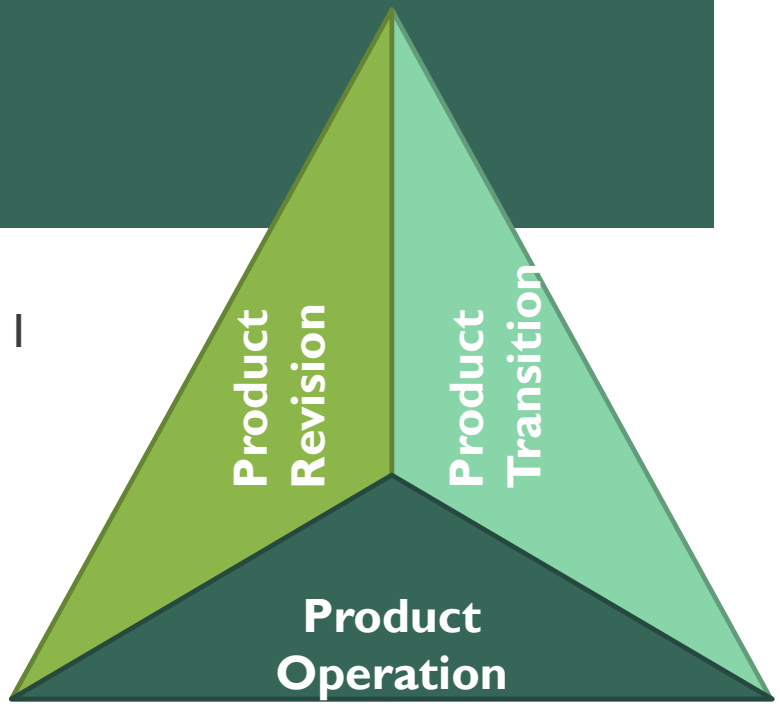
A PROCESS CAPABILITY PROFILE



MCCALL'S FACTOR MODEL

- According to this model all software requirements are classified into 11 quality factors.
- All quality factors are classified again into three groups:

Product operation factors	Product revision factors	Product transition factors
Correctness		
Reliability	Maintainability	Portability
Efficiency	Flexibility	Reusability
Integrity	Testability	Interoperability
Usability		



- McCall's quality factors were proposed in the early 1970s. They are as valid today as they were in that time.
- It's likely that software built to conform to these factors will exhibit high quality well into the 21st century, even if there are dramatic changes in technology.

PRODUCT OPERATION SOFTWARE QUALITY FACTORS

- Correctness
  - Output missing
  - Output accuracy- inaccurate data or calculations.
  - Complete output
  - Up-to-date Output
  - The availability of the information
  - The standards for coding and documenting the software system

- Reliability
  - A simple measure of reliability is *mean-time-between-failure* (MTBF), where

    MTBF = MTTF + MTTR
  - Where MTTF= *mean-time-to-failure* and

    MTTR= *mean-time-to-repair*

Availability = [MTTF/(MTTF + MTTR)] x 100%

PRODUCT OPERATION SOFTWARE QUALITY FACTORS

- Efficiency
  - Processing capabilities (given in MHz),
  - Storage capacity (given in MB or GB)
  - Data communication capability (given in MBPS or GBPS).
  - Portability of hardware
- Integrity
  - Prevent access to unauthorized persons, also to distinguish between the group of people to be given read as well as write permit.
- Usability
  - Usability requirements deal with the staff resources needed to train a new employee and to operate the software system.

PRODUCT REVISION QUALITY FACTORS

- Maintainability
  - Efforts to identify the reasons for software failures, to correct the failures, and to verify the success of the corrections.
- Flexibility
  - to support adaptive maintenance activities of the software.
- Testability
  - Testability requirements deal with the testing of the software system as well as with its operation.

PRODUCT TRANSITION SOFTWARE QUALITY FACTOR

- Portability
  - Portability requirements tend to the adaptation of a software system to other environments
- Reusability
  - This factor deals with the use of software modules originally designed for one project in a new software project currently being developed.
- Interoperability
  - Interoperability requirements focus on creating interfaces with other software systems.

SOFTWARE QUALITY METRICS

- **Software quality metrics** are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.
- Software quality metrics can be further divided into three categories −
  - **Product quality metrics**
  - **In-process quality metrics**
  - **Maintenance quality metrics**

PRODUCT QUALITY METRICS

- This metrics include the following −

1. ***Mean Time to Failure:*** It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.
2. ***Defect Density:*** It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit.
3. ***Customer Problems:*** It measures the problems that customers encounter when using the product. The problems metric is usually expressed in terms of **Problems per User-Month (PUM)**.
   - PUM = Total Problems that customers reported (true defect and non-defect oriented problems) for a time period + Total number of license months of the software during the period
   - Where, Number of license-month of the software = Number of install license of the software × Number of months in the calculation period
   - PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

PRODUCT QUALITY METRICS

4. Customer Satisfaction
   - Customer satisfaction is often measured by customer survey data through the five-point scale −
   - Very satisfied
   - Satisfied
   - Neutral
   - Dissatisfied
   - Very dissatisfied

IN-PROCESS QUALITY METRICS

- In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes −
  - Defect density during machine testing
  - Defect arrival pattern during machine testing
  - Phase-based defect removal pattern
    - In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.
  - Defect removal effectiveness

$$DRE = \frac{Defect\ removed\ during\ a\ development\ phase}{Defects\ latent\ in\ the\ product} \times 100\%$$

MAINTENANCE QUALITY METRICS

- Although much cannot be done to alter the quality of the product during this phase, following are the fixes that can be carried out to eliminate the defects as soon as possible with excellent fix quality.
  - Fix backlog and backlog management index
    - BMI = (No. of problems closed during the month / No. of problems arrived during the month) × 100%
  - Fix response time and fix responsiveness
    - The fix response time metric is usually calculated as the mean time of all problems from open to close.
    - Short fix response time leads to customer satisfaction.
- Percent delinquent fixes
  - Percent delinquent fixes = (No. fixes that exceeded the response time criteria of severity level / No. of fixes delivered in a specified time) × 100%
- Fix quality

REFERENCES

- Chapter 16- Software Quality Assurance, "Software Engineering: A Practitioner's Approach," 7/e, by Roger S. Pressman,
- Chapter 26- Process Improvement, "Software Engineering", 9/e, by Ian Sommerville.
- https://www.tutorialspoint.com/software_quality_management/software_quality_management_metrics.htm