SOFTWARE QUALITY ASSURANCE (SQA)

SOFTWARE QUALITY

- Quality of design - degree to which the design meets the functions and features specified in the requirements model.
- Quality of conformance - degree to which the implementation follows the design, and the resulting system meets its requirements and performance goals.

user satisfaction = compliant product + good quality + delivery within budget and schedule

SOFTWARE QUALITY

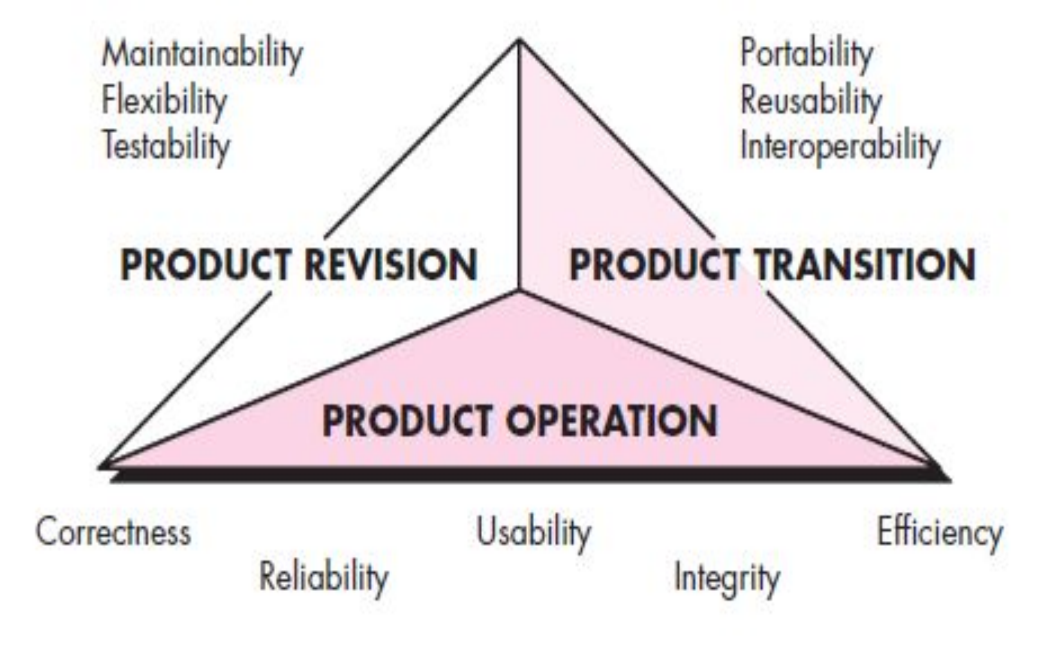
"An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it."

GARVIN'S QUALITY DIMENSIONS

1. **Performance quality**
2. **Feature quality**
3. **Reliability**
4. **Conformance**
5. **Durability**
6. **Serviceability**
7. **Aesthetics**
8. **Perception.**

MCCALL'S SOFTWARE QUALITY FACTORS

- "hard" quality factors that can be categorized in two broad groups:
- factors that can be directly measured (e.g., defects uncovered during testing)
- factors that can be measured only indirectly (e.g., usability or maintainability).



ISO 9126 QUALITY FACTORS

- **Functionality**
- **Reliability**
- **Usability**
- **Efficiency**
- **Maintainability**
- **Portability**

ACHIEVING SOFTWARE QUALITY

- Software Engineering Methods
- Project Management Techniques
- Quality Control
- Quality Assurance

SQA

- explicitly define what is meant when you say “software quality,”
- create a set of activities that will help ensure that every software engineering work product exhibits high quality
- Perform quality control and assurance activities on every software project
- use metrics to develop strategies for improving your software process

- **Who?** Everyone is responsible for quality.
- **Why?** It reduces the amount of rework that it must do.
- results in lower costs and improved time-to-market.
- **When?** it proceed with every steps of software development

SQA GROUP

- customer’s in-house representative

ELEMENTS OF SOFTWARE QUALITY ASSURANCE

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

SQA TASKS, GOALS, AND METRICS

- Tasks-
  - SQA Plan,
  - software process description,
  - review software engineering activities,
  - deviation documentation,
  - Track non-compliance

GOALS-REQUIREMENTS QUALITY

Attributes	Metrics
Ambiguity	Number of ambiguous modifiers (e.g., many, large, human-friendly)
Completeness	Number of TBA, TBD
Understandability	Number of sections/subsections
Volatility	Number of changes per requirement Time (by activity) when change is requested
Traceability	Number of requirements not traceable to design/code
Model clarity	Number of UML models Number of descriptive pages per model Number of UML errors

DESIGN QUALITY

Attributes	Metric
Architectural integrity	Existence of architectural model
Component completeness	Number of components that trace to architectural model Complexity of procedural design
Interface complexity	Average number of pick to get to a typical function or content Layout appropriateness
Patterns	Number of patterns used

CODE QUALITY

Attributes	Metric
Complexity	Cyclomatic complexity
Maintainability	Design factors
Understandability	Percent internal comments Variable naming conventions
Reusability	Percent reused components
Documentation	Readability index

QC EFFECTIVENESS

Attributes	Metric
Resource allocation	Staff hour percentage per activity
Completion rate	Actual vs. budgeted completion time
Review effectiveness	See review metrics
Testing effectiveness	Number of errors found and criticality Effort required to correct an error Origin of error

STATISTICAL SQA- SIX SIGMA RULE

The Six Sigma methodology defines three core steps:

- ***Define*** customer requirements and deliverables and project goals via well defined methods of customer communication.
- ***Measure*** the existing process and its output to determine current quality performance (collect defect metrics).
- ***Analyze*** defect metrics and determine the vital few causes.
- ***Improve*** the process by eliminating the root causes of defects.
- ***Control*** the process to ensure that future work does not reintroduce the causes of defects.

OR

- ***Design*** the process to (1) avoid the root causes of defects and (2) to meet customer requirements.
- ***Verify*** that the process model will, in fact, avoid defects and meet customer requirements.

MEASURES OF RELIABILITY AND AVAILABILITY

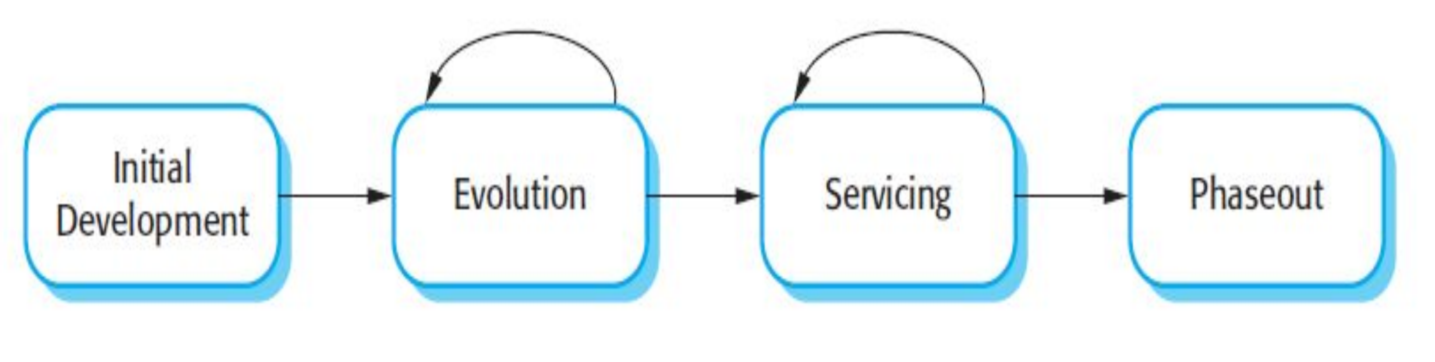
- **Software reliability -** “the probability of failure-free operation of a computer program in a specified environment for a specified time” .
- Meantime- Between-Failure (MTBF): MTBF = MTTF + MTTR

  where the acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to repair
- **Software availability-**”the probability that a program is operating according to requirements at a given point in time”

$$\text{Availability} = \frac{MTTF}{MTTF+MTTR}100\%$$

SOFTWARE EVOLUTION

- Evolution is the phase in which significant changes to the software architecture and functionality may be made.
- During servicing, the only changes that are made are relatively small, essential changes.



PROGRAM EVOLUTION DYNAMICS- LEHMAN'S LAW

- Law of Continuing Change
- Law of Increasing Complexity
- Large program evolution
- Law of Conservation of Organizational Stability
- Law of Conservation of Familiarity
- Law of Continuing Growth
- Law of Declining Quality
- Law of Feedback Systems

SOFTWARE MAINTENANCE

- Software maintenance is the general process of changing a system after it has been delivered.
- correct coding errors,
- more extensive changes to correct design errors, or
- significant enhancements to correct specification errors or
- accommodate new requirements.

TYPES OF SOFTWARE MAINTENANCE

- Fault repairs or Corrective maintenance
- Environmental adaptation or adaptive maintenance
- Functionality addition or Perfective maintenance