Software Quality Assurance and Software Reuse

Peter Lo

Software Quality

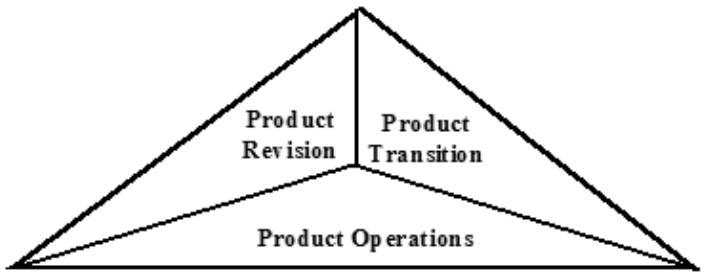
- Conformance to explicitly-stated functional and performance requirements, explicitly-documented development standards, and implicit characteristics that are expected of all professionally developed software.

Three Important Points

- Quality will be measured with Software Requirements
- Specified standards define a set of development criteria that guide the manner in which software is engineered.
- The set of implicit requirements, such as good maintainability, must still be followed.

Quality Factors

- **Product Operations:** A software product's operational characteristics
- **Product Revision:** Its ability to undergo change
- **Produced Transition:** Its adaptability to new environments



Product Operations

- Correctness
  - The extent to which a program satisfies its specification and fulfils the mission objectives.
  - Example:
    - Does it do what I want?

Product Operations

- Reliability
  - The extent to which a program can be expected to perform its intended function with required precision.
  - Example:
    - Does it do it accurately all the time?

Product Operations

- Efficiency
  - The amount of computing resources and code required by a program to perform a function.
  - Example:
    - Will it run on my hardware as well as it can?

Product Operations

- Integrity
  - The extent to which access to software or data by unauthorized persons can be controlled.
  - Example
    - Is it secure?

Product Operations

- Usability
  - The effort required to learn, operate, prepare input, and interpret output of a program.
  - Example
    - Is it designed for the user?

Product Revision

- Maintainability
  - The effort required to locate and fix an error in a program.
  - Example
    - Can I fix it?

Product Revision

- Flexibility
  - The effort required to modify an operational program.
  - Example
    - Can I change it?

Product Revision

- Testability
  - The effort required to test a program to ensure that it performs its intended function.
  - Example
    - Can I test it?

Product Transition

- Portability
  - The effort to transfer the program from one hardware and/or software system environment to another.
  - Example
    - Will I be able to use it on another machine?

Product Transition

- Reusability
  - The extent to which a program (or parts of a program) can be reused in other applications-related to the packaging and scope of the functions that the program performs.
  - Example:
    - Will I be able to reuse some of the software?

Product Transition

- Interoperability
  - The effort required to couple one system to another.
  - Example
    - Will I be able to interface it with another system?

Software Quality Assurance

- Software Quality Assurance (SQA) is a “planned and systematic pattern of actions” that are required to ensure quality in software.

SQA Activities

- Application of Technical Methods
  - Software Quality Assurance begins with a set of technical methods and tools that help the analyst to achieve a high-quality specification and the designer to develop a high-quality design.

SQA Activities

- Conduct of Formal Technical Review
  - Activity that accomplishes quality assessment for the specification and the design.
  - The Formal Technical Review is a stylized meeting conducted by technical staff with the sole purpose of uncovering quality problems.

SQA Activities

- Testing of Software
  - Combines a multi-step strategy with a series of test case design methods that help ensure effective error detection.

SQA Activities

- Enforcement of standards
  - Degree in which this is applied varies from company to company.
  - In many cases, standards are dictated by customers or regulatory mandates. In other situations, standards are self-imposed.

SQA Activities

- Control of Change
  - Every change to software has the potential of introducing errors or creating side effects that propagate errors.
  - The change control process thus contributes directly to software quality by formalizing requests for change, evaluating the nature of change, and controlling the impact of change.

SQA Activities

- Measurement
  - Track software quality and assess the impact of methodological and procedural changes on improved software quality.

SQA Activities

- Record keeping and recording
  - Collection and dissemination of SQA information.
  - The results of reviews, audits, change control, testing, and other SQA activities must become part of a historical record for a project and should be disseminated to the development staff on a need-to-know basis.

Formal Technical Reviews

- Reviews are applied at various points during software development and serve to uncover errors that can then be removed.
- Formal technical review is a class of reviews that include walkthroughs, inspections, round-robin reviews, and other small group technical assessments of software.
- It is a planned and controlled meeting attended by a group of diversified people.

Purposes of Reviews

- Point out need improvements in the product of a single person or team.
- Confirm those parts of a product in which improvement is either not desired or not needed.
- Achieve technical work of more uniform, or at least more predictable, quality than can be achieved without reviews, in order to make technical work more manageable.

Objectives of Formal Technical Review (FTR)

- To uncover errors in function, logic, or implementation for any representation of the software
- To verify that the software under review meets its requirements
- To ensure that the software has been represented according to predefined standards
- To achieve software that is developed in a uniform manner
- To make projects more manageable

Guidelines for the Organization and Preparation of FTR

- Should involve between three and five people
- Advance preparation should occur but should not require more than 2 hours of work for each person
- The duration of the review meeting should be less than 2 hours
- Focus on a components of the software
  - e.g. a portion of requirements specification, a detailed module design, a source code listing for a module
- Producer should report his/her progress

Guidelines for the Organization and Preparation of FTR

- A recorder should actively record all issues as
  - What was reviewed?
  - Who reviewed it?
  - What were the findings and conclusions?
- The attendees must make a decision at the end of the session as to
  - Accept the product
  - Reject the product
  - Accept the product provisionally, subject to further modification

Review Guidelines during the FTR

- Review the product, not the producer
- Set an agenda and maintain it
- Limit debate and rebuttal
- Enunciate problem areas, but don't attempt to solve every problem noted
- Take written notes
- Limit the number of participants and insist upon advance preparation

Review Guidelines during the FTR

- Develop a checklist for each product that is likely to be reviewed
- Allocate resources and time schedule for Formal Technical Reviews
- Conduct meaningful training for all reviewers
- Review your early reviews

Software Reliability

- **Software Reliability** is the probability of failure free operation of a computer program in a specified environment for a specified time.
- **Software Availability** is the probability that a program is operating according to requirements at a given point in time.

Types of Failure

- Transient
  - Occasional, or only with certain inputs.
- Permanent
  - Occurs with all inputs.
- Recoverable
  - Allows system to continue operation without operator intervention.

Types of Failure

- Unrecoverable
  - Forces system to halt operation.
- Non-corrupting
  - Doesn't destroy data.
- Corrupting
  - Does destroy data.

Tradeoff between Cost and Reliability

- It demands more time and resources for testing and debugging.
- It requires more internal safety checks, and this makes programs larger and slower.

Reliability Metric

- Reliability metric is an indicator of how broken a program is.
- Metrics are best weighted by the by severity of errors.
- A minor error every hour is better than a catastrophe every month.
- These metrics are not the same as counting bugs, but indicate different probabilities of happening.

Mean Time Between Failure (MTBF)

- MTBF = MTTF + MTTR
- Measures how long a program is likely to run before it does something bad like crash.
  - MTTF is the mean time to failure.
  - MTTR is the mean time to repair.

Availability

- Availability = MTTF/ (MTTF + MTTR) * 100%
  - MTTF is the mean time to failure.
  - MTTR is the mean time to repair.

Probability of Failure on Demand

- Common for safety-critical systems
  - e.g. a specification may be that there not exceed 1 chance in 10^{10} that a missile gets through.

Rate of Failure Occurrence

- This tells how many may occur in a given period
  - e.g. 2 errors per 100 minutes. This is considered one of the best metrics

Software Quality Assurance Approach

- At the low end of the scale, quality is the sole responsibility of the individual who may engineer, review, and test at any comfort level.
- At the high end of the scale, an SQA group is charged with the responsibility standards and procedures for achieving software quality and ensuring that each is followed.

Benefits of Software Quality Assurance

- Software will have fewer latent defects, resulting in reduced effort and time spent during testing and maintenance
  - Higher reliability will result in greater customer satisfaction
  - Maintenance costs
  - Overall life cycle cost of software is reduced

Constraints of Software Quality Assurance

- Difficult to institute in small organizations, where available resources to perform the necessary activities are not available
- Software Quality Assurance represents cultural change - and change is never easy
- Software Quality Assurance required the expenditure of dollars that would not otherwise be explicitly budgeted to software engineering or quality assurance

Software Reuse

- Modern, complex, high-quality computer-based systems must be built in very short time periods, and this demands a more organized approach to software reuse.

Management Issues in Software Reuse

- Few companies and organizations have anything that even slightly resembles a comprehensive software reusability plan.
- Relatively little training is available to help software engineers and managers understand and apply reuse.
- Many software practitioners continue to believe that reuse is “more trouble than it is worth.”
- Many companies continue to encourage the use of methodologies that do not facilitate reuse
- Few companies provide incentives to produce reusable program components.

Reusable Artifacts

- Project Plans
- Cost estimates
- Requirements models and specifications
- Source code
- User and technical documentation
- Human interfaces
- Data: e.g. tables, lists, record structures
- Test cases

Impact of Reuse on Quality

- Software component that is developed for reuse would contain defects.
- These defects like these can be found and eliminated, and the reused component quality improves.
- Over time, the component becomes virtually defect free.

Impact of Reuse on Productivity

- When reusable artifacts are applied throughout the software process, less time is spent creating the plans, models, documents, code and data that are normally required to create a deliverable system.
- The same level of functionality can be delivered to the customer with less input effort.

Impact of Reuse on Cost

- The net savings for reuse can be estimated by the following:
  - Net Savings = (Cost of the project developed from scratch) – (sum of costs associated with reuse) – (Actual cost of the software as delivered)