

Software Quality

The Ultimate Goal of Software Engineering



Definition of Software Quality



- Software must conform to explicit and implicit requirements if it is to be considered to be of good quality.
- Explicit requirements come in the form of
 - documented functional and performance requirements
 - documented development standards

Definition of Implicit Requirements



- Those characteristics of software that everyone expects of all professionally developed software.
- These expectations are rarely, if ever, formally documented.
- Testing for these requirements will be difficult because it is necessary to search for something that might not be there and nobody has documented that it must be there!

A Definition of Quality

A definition of quality should emphasize three important points.



Quality Definition #1



- Software requirements are the foundation for everything else in the development process.
 - Without a stated goal state, one can never evaluate how closely the finished product complies with the expected product.

Verification and Validation



- Verification is the activity of evaluating whether the system is correctly built, i.e. that it fulfills its stated requirements.
- Validation is the activity of evaluating whether the right system has been built, i.e. that it fulfills the user's needs.

Quality Definition #2



- Specified standards define a set of development criteria that guide the manner in which software is developed.
 - If standards are used then they must be followed.

Quality Definition #3



- There is always a set of implicit requirements that clients/users expect but can rarely articulate during the requirements phase of development.
 - If software conforms to its explicit requirements but fails to meet implicit requirements, then it often fails to be successful in the long term (and sometimes even in the short term!).

Software Quality Models and Factors



ISO 9126 Quality Model

A standard for evaluating software quality.

Based on McCall's model of software quality.



McCall's Software Quality Model



- McCall's decompositional approach to modeling software quality was developed in 1977.
- McCall, J.A., Richards, P.K., and Walters, G.F., "Factors in Software Quality", RADDC TR-77-369, 1977, Vols I, II, III, US Rome Air Development Center Reports.
- The McCall model is an example of a fixed model approach to software quality.

Fixed Model Approach



- Assumption
 - Published model describes all quality factors needed to monitor a project and we accept the associated criteria and metrics.
- Process
 - Collect appropriate data and apply the model to determine the quality of the product.

...the other type of approach is...

Define your own Model Approach



- Assumption
 - The general philosophy that software quality can be decomposed into factors is accepted but no one model is adopted or followed completely.
- Process
 - Meet with users/clients and reach a consensus on those quality factors that are important for your particular project and on the criteria and metrics and their relationship to quality.

McCall's Software Quality Model



- There were three levels of decomposition in McCall's model:
 - 1) Use
 - 2) Factor
 - factors that can be directly measured
 - factors that can be measured only indirectly (e.g. usability and maintainability)
 - 3) Criteria
 - All criteria have a corresponding metric(s).

The Top Level



- Use focused on three important aspects
 - a) Product operation
 - b) Product revision
 - c) Product transition

Product Operation



- Usability
- Integrity
- Efficiency
- Correctness
- Reliability

Product Revision



- Maintainability
- Testability
- Flexibility

Product Transition



- Reusability
- Portability
- Interoperability

ISO 9126



- McCall model was used in 1992 as the basis for an international standard.
- ISO 9126 (ISO 1991) is called Software Product Evaluation: Quality Characteristics and Guidelines for their Use.

Software Quality Definition



- In this standard, software quality is defined to be

“the totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs.”

ISO 9126 Software Quality Factors



- Quality is decomposed into six factors
 1. Functionality
 2. Reliability
 3. Efficiency
 4. Usability
 5. Maintainability
 6. Portability

Functionality



- The software performs as per the requirements and specifications.
- Testing is used to verify that the requirements are met.
- Obviously this is the most basic of quality factors but can be problematic for large, complex software.

Reliability



- Reliability is the capability of software to maintain its level of performance under stated conditions for a stated period of time. It is also defined as the probability of failure-free operation.
- Criteria include
 - Accuracy
 - Error tolerance
 - Consistency
 - Simplicity

Efficiency



- We can measure efficiency indirectly by measuring the amount of time (execution efficiency) or storage (storage efficiency) needed when running the software through a particular compiler, under a specific OS, on a designated hardware architecture.
- Efficiency can be better understood by abstracting the software and subjecting it to measurement studies in a formal sense.

Usability



- Usability is an attempt to define user friendliness.
- It can be measured in terms of:
 - physical and intellectual skill required to learn the system
 - time required to become moderately efficient in the use of the system
 - the net increase in productivity over the system it replaces
 - a subjective assessment of users' attitudes towards the system

Maintainability



- Software has a high degree of maintainability if it is easy to understand, enhance, and correct.
- Criteria include consistency, simplicity, conciseness, self-descriptiveness, and modularity.
- Maintainability cannot be measured directly.
 - MTTC (mean time to change) is the time it takes to analyze, design, and implement the change. Maintainable programs have a lower MTTC.

Portability



- Portability is a set of attributes that bear on the capability of software to be transferred from one environment to another.
- Criteria
 - Self-descriptiveness
 - Modularity
 - Machine independence
 - Software system independence

What was left out?



- The factors in the McCall model that are not part of the basic set in ISO 9126 are:
 - Integrity
 - Correctness
 - Testability
 - Flexibility
 - Reusability
 - Interoperability

Integrity



- Integrity is the ability of a system to withstand attacks to its security.
- Criteria include access control and access audit.
- To measure integrity one must define:
 - a) Threat: the probability that an attack will occur within a given time.
 - b) Security: the probability that the attack of a specific type will be repelled.

Correctness



- Correctness is the degree to which software performs its desired function.
- A common measure of correctness is defects per KLOC.
- Defects are caught:
 - during regression testing
 - reported by the user during beta testing or in regular use of the product

Testability



- Testability can be defined as the probability that software will fail if it is faulty
- Criteria
 - Simplicity
 - Instrumentation
- Important testing measurements include
 - Minimum number of test cases
 - Test effectiveness ratio

Flexibility



- Flexibility describes the ease with which the software can be used in various situations and environments.
- Criteria
 - Expandability
 - Generality
 - Self-descriptiveness
 - Modularity

Reusability

- Reusability is the ability of components of the software to be used in other applications.
- Criteria
 - Generality
 - Self-descriptiveness
 - Modularity
 - Machine independence
 - Software system independence

Interoperability

- Interoperability is the ability of the software to work with other software systems or to co-exist without causing difficulties.
- Criteria
 - Modularity
 - Communications commonality
 - Data commonality

ISO 9126 as a Standard

- ISO 9126 claims that these are comprehensive and each can be refined through multiple levels of subcharacteristics.
- The standard does not, however, define attributes at the second level of refinement.
- This has led to criticism of the standard as a useful mechanism for monitoring quality but ISO 9126 is an important milestone in quality measurement.

ISO 9126 Evaluation Process Model

