**Managing Software Quality**

**Software quality management** (**SQM**) is a management process that aims to develop and manage the quality of software in such a way so as the best ensure the product meets the quality standards expected by the customer while also meeting any necessary regulatory and developer requirements, if any. Software quality managers require software to be tested before it is released to the market, and they do this using a cyclical process-based quality assessment in order to reveal and fix bugs before release. Their job is not only to ensure their software is in good shape for the consumer but also to encourage a culture of quality throughout the enterprise.

Software quality management activities are generally split up into three core components: quality assurance, quality planning, and quality control.[[1]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-ZsoltSoftware14-1)[[2]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-MaximSoftware14-2) Some like software engineer and author Ian Sommerville don't use the term "quality control" (as quality control is often viewed as more a manufacturing term than a software development term), rather, linking its associated concepts with the concept of quality assurance.[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3) However, the three core components otherwise remain the same.

**Quality assurance** [

By setting up an organized and logical set of organizational processes and deciding on that software development standards — based on industry best practices — that should be paired with those organizational processes, software developers stand a better chance of producing higher quality software. However, linking quality attributes such as "maintanability" and "reliability" to processes is more difficult in software development due to its creative design elements versus the mechanical processes of manufacturing.[ Additionally, "process standardization can sometimes stifle creativity, which leads to poorer rather than better quality software."[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3)

This stage can include:

* encouraging documentation process standards, such as the creation of well-defined engineering documents using standard templates
* mentoring how to conduct standard processes, such as quality reviews
* performing in-process test data recording procedures
* identifying standards, if any, that should be used in software development processes

**Quality planning**[

Quality planning works at a more granular, project-based level, defining the quality attributes to be associated with the output of the project and how those attributes should be assessed. Additionally, any existing organizational standards may also be assigned to the project at this phase. Attributes such as "robustness," "accessibility," and "modularity" may be assigned to the software development project.[[1]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-ZsoltSoftware14-1)[[2]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-MaximSoftware14-2) While this may be a more formalized, integral process, those using a more agile method of quality management may place less emphasis on strict planning structures.[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3) The quality plan may also address intended market, critical release dates, quality goals, expected risks, and risk management policy.[[2]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-MaximSoftware14-2)[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3)

**Quality control**[]

The quality control team tests and reviews software at its various stages to ensure quality assurance processes and standards at both the organizational and project level are being followed.[[1]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-ZsoltSoftware14-1)[[2]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-MaximSoftware14-2) (Some like Sommerville link these responsibilities to quality assurance rather than call it quality control.[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3)) These checks are optimally separate from the development team so as to lend more of an objective view of the product to be tested.[[2]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-MaximSoftware14-2)[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3) However, project managers on the development side must also assist, helping to promote as part of this phase a "culture that provides support without blame when errors are discovered."[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3) In software development firms implementing a more agile quality approach, these activities may be less formal; however, a switch to agile methods from a more formal quality management structure may create problems if management procedures aren't appropriately adapted.[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3)

Activities include:

* release testing of software, including proper documentation of the testing process
* examination of software and associated documentation for non-conformance with standards
* follow-up review of software to ensure any required changes detailed in previous testing are addressed
* application of software measurement and metrics for assessment

Software quality and the software life cycle

The measurement of software quality is different from manufacturing; tolerances aren't applicable (at least in the same way), and objective conclusions concerning if software meets specifications are difficult if not impossible to achieve.[[3]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-SommervilleSoftware11-3) However, software's quality and fit-for-purpose status can still be realized in various ways depending on the organization and type of realized project.[[4]](https://en.wikipedia.org/wiki/Software_quality_management#cite_note-4) This done through the support of the entire **software development lifecycle,** meaning:

* collecting requirements and defining the scope of an IT project, focused on verification if defined requirements will be testable;
* designing the solution, focused on planning a test process, e.g., what type of tests will be performed and how they will be performed in the context of test environments and test data?;
* implementing a solution supported by test cases and scenarios, executing them, and registering defects, including the coordination of resolving the defects;
* implementing change management, supported by verification of how planned changes can influence the quality of a created solution and eventual change of a test plan; and
* closing the project, supported by the realization of tests focused on complex verification of the overall quality of the created solution.

**Software Quality**

Quality software refers to a software which is reasonably bug or defect free, is delivered in time and within the specified budget, meets the requirements and/or expectations, and is maintainable. In the software engineering context, software quality reflects both **functional quality** as well as **structural quality**.

* **Software Functional Quality** − It reflects how well it satisfies a given design, based on the functional requirements or specifications.
* **Software Structural Quality** − It deals with the handling of non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, and the degree to which the software was produced correctly.
* **Software Quality Assurance** − Software Quality Assurance (SQA) is a set of activities to ensure the quality in software engineering processes that ultimately result in quality software products. The activities establish and evaluate the processes that produce products. It involves process-focused action.
* **Software Quality Control** − Software Quality Control (SQC) is a set of activities to ensure the quality in software products. These activities focus on determining the defects in the actual products produced. It involves product-focused action.

## The Software Quality Challenge

In the software industry, the developers will never declare that the software is free of defects, unlike other industrial product manufacturers usually do. This difference is due to the following reasons.

### Product Complexity

It is the number of operational modes the product permits. Normally, an industrial product allows only less than a few thousand modes of operation with different combinations of its machine settings. However, software packages allow millions of operational possibilities. Hence, assuring of all these operational possibilities correctly is a major challenge to the software industry.

### Product Visibility

Since the industrial products are visible, most of its defects can be detected during the manufacturing process. Also the absence of a part in an industrial product can be easily detected in the product. However, the defects in software products which are stored on diskettes or CDs are invisible.

### Product Development and Production Process

In an industrial product, defects can be detected during the following phases −

* **Product development** − In this phase, the designers and Quality Assurance (QA) staff checks and tests the product prototype to detect its defects.
* **Product production planning** − During this phase, the production process and tools are designed and prepared. This phase also provides opportunities to inspect the product to detect the defects that went unnoticed during the development phase.
* **Manufacturing** − In this phase, QA procedures are applied to detect failures of products themselves. Defects in the product detected in the first period of manufacturing can usually be corrected by a change in the product’s design or materials or in the production tools, in a way that eliminates such defects in products manufactured in future.

However, in the case of software, the only phase where defects can be detected is the development phase. In case of software, product production planning and manufacturing phases are not required as the manufacturing of software copies and the printing of software manuals are conducted automatically.

he various factors, which influence the software, are termed as software factors. They can be broadly divided into two categories. The first category of the factors is of those that can be measured directly such as the number of logical errors, and the second category clubs those factors which can be measured only indirectly. For example, maintainability but each of the factors is to be measured to check for the content and the quality control.

Several models of software quality factors and their categorization have been suggested over the years. The classic model of software quality factors, suggested by McCall, consists of 11 factors (McCall et al., 1977). Similarly, models consisting of 12 to 15 factors, were suggested by Deutsch and Willis (1988) and by Evans and Marciniak (1987).

All these models do not differ substantially from McCall’s model. The McCall factor model provides a practical, up-to-date method for classifying software requirements (Pressman, 2000).

## McCall’s Factor Model

his model classifies all software requirements into 11 software quality factors. The 11 factors are grouped into three categories – product operation, product revision, and product transition factors.

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

## Product Operation Software Quality Factors

According to McCall’s model, product operation category includes five software quality factors, which deal with the requirements that directly affect the daily operation of the software. They are as follows −

### Correctness

These requirements deal with the correctness of the output of the software system. They include −

* Output mission
* The required accuracy of output that can be negatively affected by inaccurate data or inaccurate calculations.
* The completeness of the output information, which can be affected by incomplete data.
* The up-to-dateness of the information defined as the time between the event and the response by the software system.
* The availability of the information.
* The standards for coding and documenting the software system.
* Reliability
* Reliability requirements deal with service failure. They determine the maximum allowed failure rate of the software system, and can refer to the entire system or to one or more of its separate functions.
* Efficiency
* It deals with the hardware resources needed to perform the different functions of the software system. It includes processing capabilities (given in MHz), its storage capacity (given in MB or GB) and the data communication capability (given in MBPS or GBPS).
* It also deals with the time between recharging of the system’s portable units, such as, information system units located in portable computers, or meteorological units placed outdoors.
* Integrity
* This factor deals with the software system security, that is, to 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
* According to McCall’s model, three software quality factors are included in the product revision category. These factors are as follows −
* Maintainability
* This factor considers the efforts that will be needed by users and maintenance personnel to identify the reasons for software failures, to correct the failures, and to verify the success of the corrections.
* Flexibility
* This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software. These include adapting the current software to additional circumstances and customers without changing the software. This factor’s requirements also support perfective maintenance activities, such as changes and additions to the software in order to improve its service and to adapt it to changes in the firm’s technical or commercial environment.
* Testability
* Testability requirements deal with the testing of the software system as well as with its operation. It includes predefined intermediate results, log files, and also the automatic diagnostics performed by the software system prior to starting the system, to find out whether all components of the system are in working order and to obtain a report about the detected faults. Another type of these requirements deals with automatic diagnostic checks applied by the maintenance technicians to detect the causes of software failures.
* Product Transition Software Quality Factor
* According to McCall’s model, three software quality factors are included in the product transition category that deals with the adaptation of software to other environments and its interaction with other software systems. These factors are as follows −
* Portability
* Portability requirements tend to the adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth. The software should be possible to continue using the same basic software in diverse situations.
* Reusability
* This factor deals with the use of software modules originally designed for one project in a new software project currently being developed. They may also enable future projects to make use of a given module or a group of modules of the currently developed software. The reuse of software is expected to save development resources, shorten the development period, and provide higher quality modules.
* Interoperability
* Interoperability requirements focus on creating interfaces with other software systems or with other equipment firmware. For example, the firmware of the production machinery and testing equipment interfaces with the production control software.

**Software Quality Assurance** (SQA) is a set of activities for ensuring quality in software engineering processes. It ensures that developed software meets and complies with the defined or standardized quality specifications. SQA is an ongoing process within the Software Development Life Cycle (SDLC) that routinely checks the developed software to ensure it meets the desired quality measures.

SQA practices are implemented in most types of software development, regardless of the underlying software development model being used. SQA incorporates and implements software testing methodologies to test the software. Rather than checking for quality after completion, SQA processes test for quality in each phase of development, until the software is complete. With SQA, the software development process moves into the next phase only once the current/previous phase complies with the required quality standards. SQA generally works on one or more industry standards that help in building software quality guidelines and implementation strategies.

It includes the following activities −

* Process definition and implementation
* Auditing
* Training

Processes could be −

* Software Development Methodology
* Project Management
* Configuration Management
* Requirements Development/Management
* Estimation
* Software Design
* Testing, etc.

Once the processes have been defined and implemented, Quality Assurance has the following responsibilities −

* Identify the weaknesses in the processes
* Correct those weaknesses to continually improve the process

Components of SQA System

An SQA system always combines a wide range of SQA components. These components can be classified into the following six classes −

Pre-project components

This assures that the project commitments have been clearly defined considering the resources required, the schedule and budget; and the development and quality plans have been correctly determined.

Components of project life cycle activities assessment

The project life cycle is composed of two stages: the development life cycle stage and the operation–maintenance stage.

The development life cycle stage components detect design and programming errors. Its components are divided into the following sub-classes: Reviews, Expert opinions, and Software testing.

The SQA components used during the operation–maintenance phase include specialized maintenance components as well as development life cycle components, which are applied mainly for functionality to improve the maintenance tasks.

Components of infrastructure error prevention and improvement

The main objective of these components, which is applied throughout the entire organization, is to eliminate or at least reduce the rate of errors, based on the organization’s accumulated SQA experience.

Components of software quality management

This class of components deal with several goals, such as the control of development and maintenance activities, and the introduction of early managerial support actions that mainly prevent or minimize schedule and budget failures and their outcomes.

Components of standardization, certification, and SQA system assessment

These components implement international professional and managerial standards within the organization. The main objectives of this class are utilization of international professional knowledge, improvement of coordination of the organizational quality systems with other organizations, and assessment of the achievements of quality systems according to a common scale. The various standards may be classified into two main groups: quality management standards and project process standards.

Organizing for SQA – the human components

The SQA organizational base includes managers, testing personnel, the SQA unit and the persons interested in software quality such as SQA trustees, SQA committee members, and SQA forum members. Their main objectives are to initiate and support the implementation of SQA components, detect deviations from SQA procedures and methodology, and suggest improvements.

Pre-project Software Quality Components

These components help to improve the preliminary steps taken before starting a project. It includes −

* Contract Review
* Development and Quality Plans

Contract Review

Normally, a software is developed for a contract negotiated with a customer or for an internal order to develop a firmware to be embedded within a hardware product. In all these cases, the development unit is committed to an agreed-upon functional specification, budget and schedule. Hence, contract review activities must include a detailed examination of the project proposal draft and the contract drafts.

Specifically, contract review activities include −

* Clarification of the customer’s requirements
* Review of the project’s schedule and resource requirement estimates
* Evaluation of the professional staff’s capacity to carry out the proposed project
* Evaluation of the customer’s capacity to fulfil his obligations
* Evaluation of development risks

Development and Quality Plans

After signing the software development contract with an organization or an internal department of the same organization, a development plan of the project and its integrated quality assurance activities are prepared. These plans include additional details and needed revisions based on prior plans that provided the basis for the current proposal and contract.

Most of the time, it takes several months between the tender submission and the signing of the contract. During these period, resources such as staff availability, professional capabilities may get changed. The plans are then revised to reflect the changes that occurred in the interim.

The main issues treated in the project development plan are −

* Schedules
* Required manpower and hardware resources
* Risk evaluations
* Organizational issues: team members, subcontractors and partnerships
* Project methodology, development tools, etc.
* Software reuse plans

The main issues treated in the project’s quality plan are −

* Quality goals, expressed in the appropriate measurable terms
* Criteria for starting and ending each project stage
* Lists of reviews, tests, and other scheduled verification and validation activities

oftware metrics can be classified into three categories −

* **Product metrics** − Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
* **Process metrics** − These characteristics can be used to improve the development and maintenance activities of the software.
* **Project metrics** − This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project 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 −

* Mean Time to Failure
* Defect Density
* Customer Problems
* Customer Satisfaction

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.

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. This metric is used in many commercial software systems.

Customer Problems

It measures the problems that customers encounter when using the product. It contains the customer’s perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

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.

Customer Satisfaction

Customer satisfaction is often measured by customer survey data through the five-point scale −

* Very satisfied
* Satisfied
* Neutral
* Dissatisfied
* Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys. Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis. For example −

* Percent of completely satisfied customers
* Percent of satisfied customers
* Percent of dis-satisfied customers
* Percent of non-satisfied customers

Usually, this percent satisfaction is used.

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
* Defect removal effectiveness

Defect density during machine testing

Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.

Defect arrival pattern during machine testing

The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following −

* The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.
* The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.
* The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

Phase-based defect removal pattern

This is an extension of the defect density metric during testing. 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.

Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software. The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

Defect removal effectiveness

It can be defined as follows −

DRE=DefectremovedduringadevelopmentphaseDefectslatentintheproduct×100%DRE=DefectremovedduringadevelopmentphaseDefectslatentintheproduct×100%

This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called **early defect removal** when used for the front-end and **phase effectiveness** for specific phases. The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

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
* Fix response time and fix responsiveness
* Percent delinquent fixes
* Fix quality

Fix backlog and backlog management index

Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.

Backlog Management Index (BMI) is used to manage the backlog of open and unresolved problems.

BMI=NumberofproblemsclosedduringthemonthNumberofproblemsarrivedduringthemonth×100%BMI=NumberofproblemsclosedduringthemonthNumberofproblemsarrivedduringthemonth×100%

If BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

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.

The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.

Percent delinquent fixes

It is calculated as follows −

PercentDelinquentFixes=PercentDelinquentFixes=

NumberoffixesthatexceededtheresponsetimecriteriabyceveritylevelNumberoffixesdeliveredinaspecifiedtime×100%NumberoffixesthatexceededtheresponsetimecriteriabyceveritylevelNumberoffixesdeliveredinaspecifiedtime×100%

**Total Quality Management (TQM)**

TQM can be defined as a management technique for improving processes, products, services and the other approaches associated with the product. It focusses on the entire business and NOT just on a particular project or process.

## Elements of TQM:

* Root Cause Analysis
* Customer-focused
* Active Employee Participation
* Process-oriented
* Internal and External self Assessment
* Continuous improvement
* Making Well Informed Decisions
* Effective Communication

## Quality Control Tools:

* Cause - Effect Diagram
* Checklists
* Histogram
* Graphs
* Pareto Charts
* Tree Diagram
* Arrow Diagram

Process Improvement Cycle:

