Software Quality

**Robert Hughes and
Mike Cotterell**

**Department of Computer Science , National Textile University Faisalabad-37610, Pakistan
Office Phone: +92-41-9230081 Ext: 119 | Fax: +92 (41) 9230098 | email: cmn.faisal@ntu.edu.pk**

Software Quality

Objective

- Explain the importance of software quality to software users and software developers
- Define the qualities of a good software
- Design methods for measuring the required qualities of software
- Monitor the quality of the processes in a software project
- Use external quality standards to ensure the quality of software acquired from an outside supplier
- Develop system using procedures that will increase their quality

12.2 Introduction

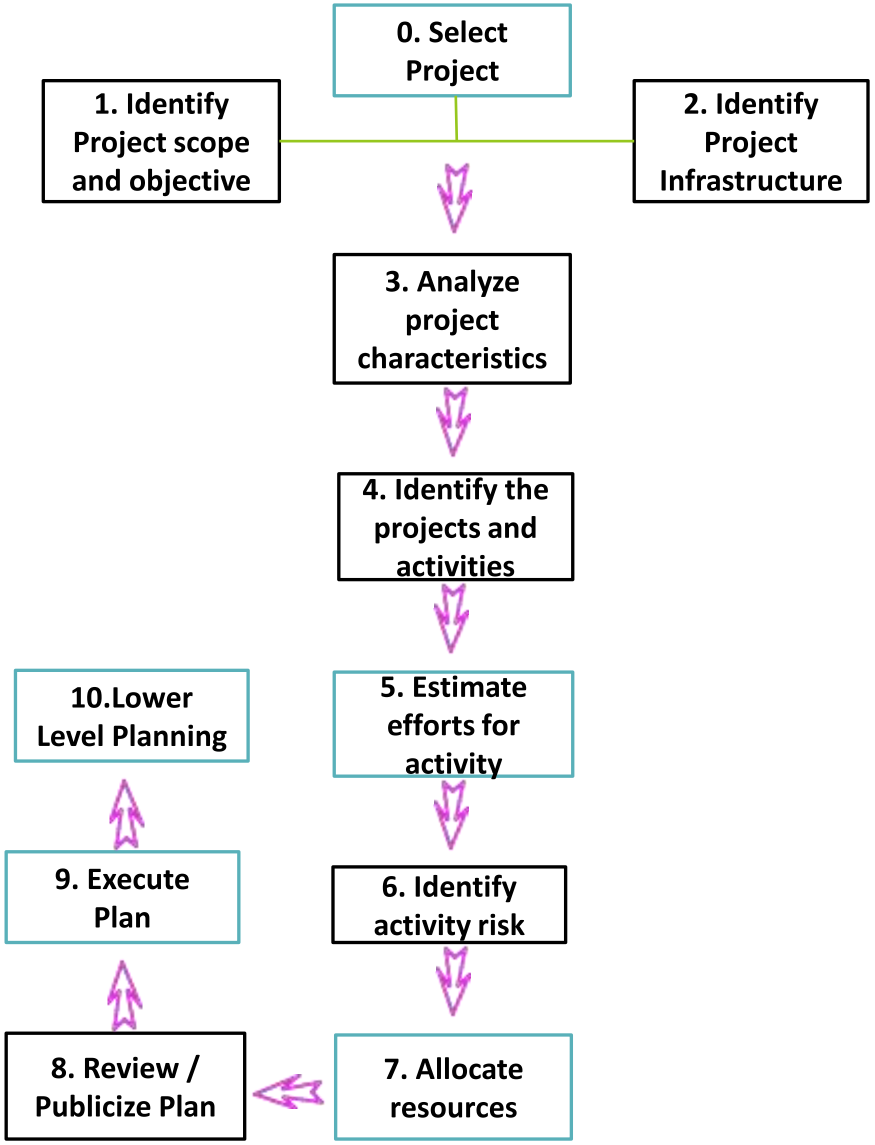
Quality is generally agreed to be a good thing. In a practice the quality of a system can be a vague, undefined, attribute. We therefore need to define precisely what qualities were require of a system. However, this is not enough – we need to objectively weather a system meet out quality requirements and this need measurement.

Software Quality

12.2 Place of software quality in project management.

Quality will be of concern at all stages of project planning and execution. But will be of particular interest at the following points in the step wise framework

Software Quality


0. Select Project
1. Identify Project scope and objective
2. Identify Project Infrastructure
3. Analyze project characteristics
4. Identify the projects and activities
5. Estimate efforts for activity
6. Identify activity risk
7. Allocate resources
8. Review / Publicize Plan
9. Execute Plan
10.Lower Level Planning

Software Quality

1. Identify Project Scope and objective: Some objective could relate to the qualities of the application to be delivered.

2. Identify project infrastructure: Identify the installation standard and procedures. Some of these almost certainly be about quality

3. Analyze project characteristics: To identify the other qualities based requirement.

4. Identify the products and activities of the project: It is at this point the entry, exist and process requirement are identified for each activity

5. Review and publicize Plan: At his stage the overall quality aspects of the project plan are reviewed

Software Quality

12.3 Importance of Software Quality

We would expect quality to be concern of all procedures of goods and services.

- Increasing criticality of software: the end user of a software generally anxious about the quality of software especially about the reliability. They are concern about the safety because of their dependency on the software system such as aircraft control system are more safety critical systems

- The intangibility of software: This make it difficulty to know that a particular tasks in project has been completed satisfactory. The results of these tasks can be made tangible by demanding that the developer produce deliverables that can be examined for quality

Software Quality

- Accumulating errors during software development:

As computer system developed in made up of a number of steps where the output from one step is the input to the next, the error in the earlier deliverables will be added to those in the later steps leading to an accumulating determined effects. In general the later in a project that an error is found the more expensive it will be to fix. In addition because the number of errors in the system is unknown, the debugging phases of a project are particularly difficult to control

12.4 Defining the software Quality:

Quality is a rather vague term and we need to define carefully what we mean by it.

- A functional specification describing what the system is to do
- A quality specification concerned with how well the functions are to operate
- A resource specification concerned with how much in to be spent on the system

Software Quality

Attempt to identify specific product qualities that are appropriate to software, for instance, grouped software qualities into three sets. Product operation qualities, Products revision qualities and product transition qualities.

Product operation qualities

Correctness: The extent to which a program satisfy its specification and fulfil user objective

Reliability: The extent to which a program can be expected to perform its intended function with required precision

Efficiency: The amounts of computer resource required by software

Integrity: The extent to which access to software or data by unauthorized persons can be controlled

Usability: The effort required to learn, operate, prepare input and interprets output

Software Quality

Product revision qualities

Maintainability: the effort required to locate and fix an error in an operational program

Testability: The effort required to test a program to ensure it performs its intended function

Flexibility: The effort required to modify an operational program,

Product Transition qualities

Portability: The efforts required to transfer a program from one hardware configuration and or software system environment to another

Reusability: The extent to which a program can be used in other applications.

Interoperability: The efforts required to couple one system to another

Software Quality

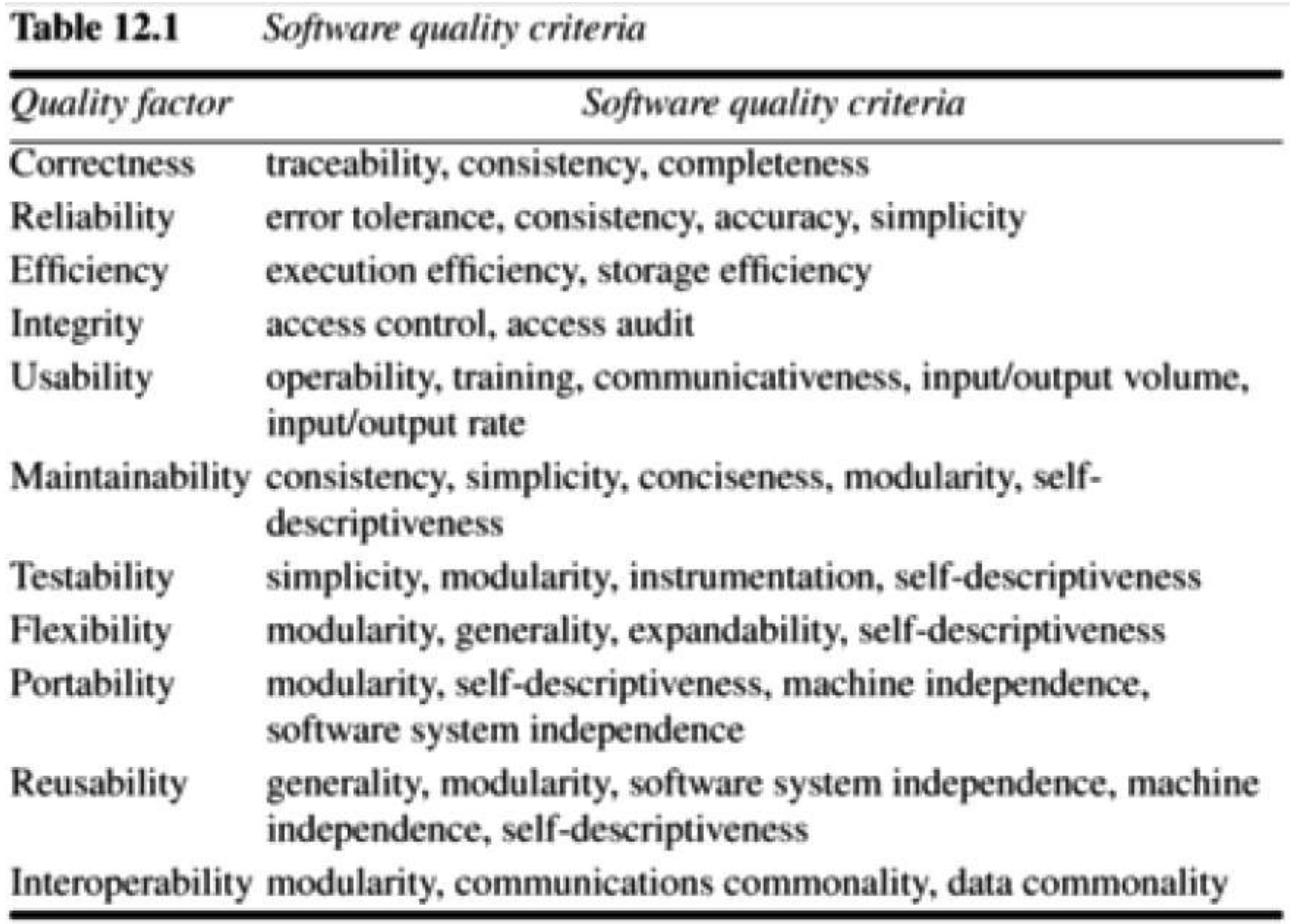


Table 12.1 *Software quality criteria*

Quality factor	*Software quality criteria*
Correctness	traceability, consistency, completeness
Reliability	error tolerance, consistency, accuracy, simplicity
Efficiency	execution efficiency, storage efficiency
Integrity	access control, access audit
Usability	operability, training, communicativeness, input/output volume, input/output rate
Maintainability	consistency, simplicity, conciseness, modularity, self-descriptiveness
Testability	simplicity, modularity, instrumentation, self-descriptiveness
Flexibility	modularity, generality, expandability, self-descriptiveness
Portability	modularity, self-descriptiveness, machine independence, software system independence
Reusability	generality, modularity, software system independence, machine independence, self-descriptiveness
Interoperability	modularity, communications commonality, data commonality

Software Quality

During quality is not enough. If we are to judge whether a system meets out requirements we need to be able to measure its qualities. For each criterion, one or more measure have to be invented the degree to which the quality is present. In general the user of software would be concerned with measuring what McCall called quality factors while the developers would be concerned with quality criteria. The following should be laid down for each quality.

Scale – the unit of measurement

Test – the practical test of the extent to which the attribute quality exists

Worst – the worst acceptable value

Plan – the value that is planned to achieve

Best – the best value that appears to be feasible

Now – the values that applies currently

Software Quality

12.5 ISO-9126:

ISO 9126 standard was published in 1991 to tackle the question of the definition of software quality this 13 pages document was designed as foundation upon which further, more detailed standard could be built.

ISO 9126 identifies six software quality characteristics

- Functionality: which covers the functions that a software product provides to satisfy user needs
- Reliability: Which relates to the capability of the software to maintain its level of performance
- Usability: Efforts need to use a software
- Efficiency: physical resource used when a software is executed
- Maintainability: Effort needed to the make changes to the software
- Portability: Availability of the software to be transferred to a different environment.

Software Quality

Functionality: which covers the functions that a software product provides to satisfy user needs.

Functionality sub characteristics: suitability, Accuracy, Interoperability, Compliance and Security.

Compliance refers to the degree to which the software adheres to application-related standards or legal requirements. Typically these could be auditing requirement. Interoperability refers to the ability of software to interact with others.

Reliability: Which relates to the capability of the software to maintain its level of performance

Reliability sub characteristics: Maturity, Fault Tolerance and recoverability.

Maturity refers to frequency of failures due to fault in software more identification of fault more chances to remove them. Recoverability describe the control of access to a system

Software Quality

Usability: Efforts need to use a software.

Usability sub characteristics: Understand ability, Learnability, operability.

Understand-ability is a clear quality to grasp, although the definition attributes that bear on the user efforts for recognizing the logical concept and its applicability in our view actually makes it less clear. Learnability has been distinguished from operability. A software tool might be easy to learn but time-consuming to use say it uses a large number of nested menus. This is for a package that is used only intermittently but not where the system is used or several hours each day by the end user. In this case learnability has been incorporated at the expense of operability.

Software Quality

Efficiency: physical resource used when a software is executed

Efficiency sub characteristics: Time behaviour, Resource behaviour.

Maintainability: Effort needed to the make changes to the software

Maintainability sub characteristics: Analysability, Changeability, Stability and Testability.

Analysability is the quality that McCall called diagnose ability, the ease with which the cause of failure can be determined. Changeability is the quality that other have called flexibility: the latter name is perhaps a better one as changeability has a slightly different connotation in plain English It implies that the suppliers of the software are always changing it. Stability, on the other hand, does not means that the software never changes, It means that there is a low risk of a modification to the software having unexpected effects.

Software Quality

Portability: Availability of the software to be transferred to a different environment.

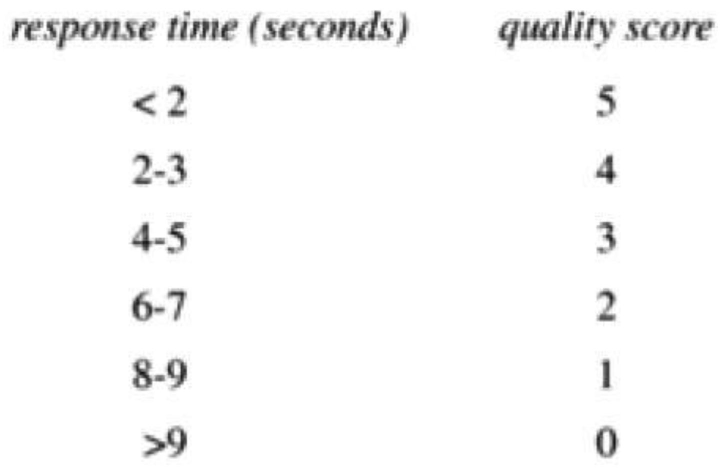
Portability sub characteristics: Adaptability, Install ability , Conformance and Replace ability.

Conformance is distinguished from compliance relates to those standard that have bearing on portability. The use of a standard programming language common to many software/hardware environment is an example of conformance. Replace ability refers to the factors that give upwards compatibility between old software components and the new ones. Downwards compatibility is specifically excluded from definition.

Software Quality

Quality Metrics selection: Measurements that correlate to the characteristics of each quality have to be identified. No specific guidance is given by ISO 9129 standard on the applicability of the various measurements that might be used.

Rating Level Definition: The metrics used must be mapped onto scales that indicate the degree to which the requirement have been satisfied for example in one application time behaviour in the sense of response time might be important for a key transaction actual response time might be mapped onto quality scale.



response time (seconds)	*quality score*
< 2	5
2-3	4
4-5	3
6-7	2
8-9	1
>9	0

Software Quality

Assessment criteria definitions: The way that the quality scores are combined or summarized or give an overall view of the product has to be defined. There software product as now to be evaluated by measuring its qualities, converting them to quality score or rating and summarising them the rating to obtain an overall judgment.

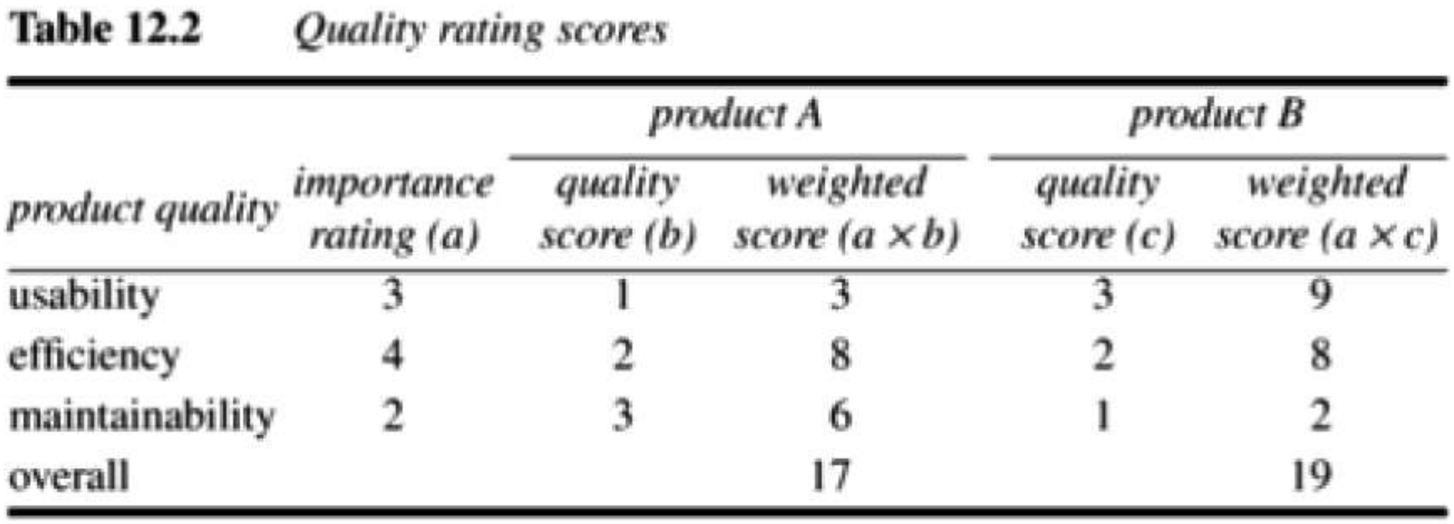


Table 12.2 *Quality rating scores*

		product A		*product B*	
product quality	*importance rating (a)*	*quality score (b)*	*weighted score (a × b)*	*quality score (c)*	*weighted score (a × c)*
usability	3	1	3	3	9
efficiency	4	2	8	2	8
maintainability	2	3	6	1	2
overall			17		19

Software Quality

Practical software quality measures : Below are some way of measuring particular qualities.

Reliability: might be measure in terms of

Availability: the percentage of a particular time interval that a system is usable.

Means time between failures, the total service time divided by the number of failures

Failure on demand: the probability that a system will not be available at the time required on the probability that a transaction will fail.

Support activity: the number of fault reports that are dealt with

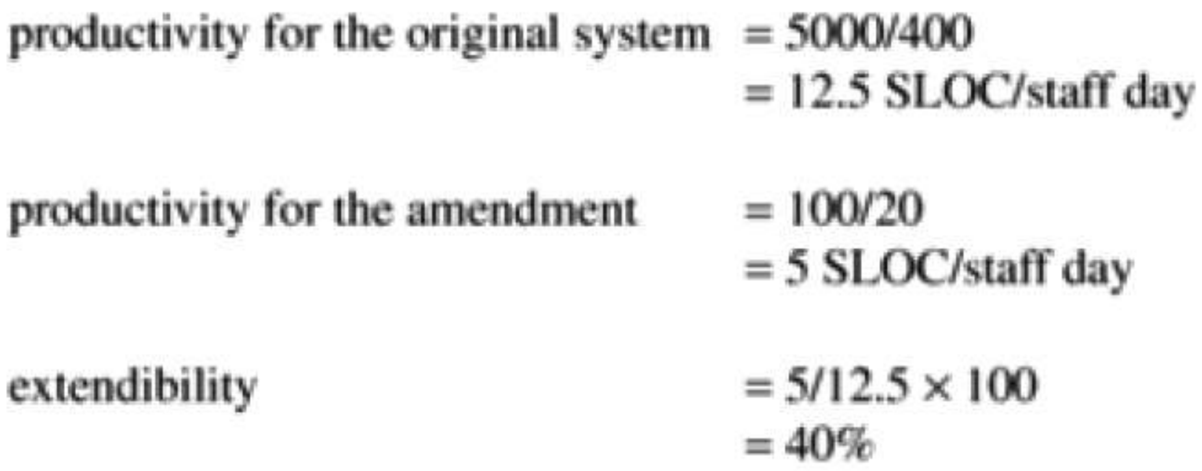
Software Quality

Maintainability: This is closely related to flexibility the ease with which the software can be modified. The main deference is that before an amendment can be made, the fault has to be diagnosed. Maintainability can therefore be seen as flexibility plus a new quality, diagnose ability which might be defined as the average amount of time needed to diagnose a fault.

Extendibility: This is a component of the more general quality of flexibility. It can be defined as the productivity needed to incorporate a new feature into an existing system expressed as a percentage of the normal productivity when developing the software from scratch.

Software Quality

The original IOE maintenance billing system comprised 5000 SLOC and took 400 works-days to implement. An amendment t the core system caused by the introduction of group accounts has lead to 100 SLOC being added which took 20 works days to implements thus



productivity for the original system = 5000/400
= 12.5 SLOC/staff day

productivity for the amendment = 100/20
= 5 SLOC/staff day

extendibility = $5/12.5 \times 100$
= 40%

Software Quality

12.7 Product Versus Process Quality Management

The measurement describe above can be taken only after the system is operational. It might be too late to do anything to remedy problems. What would be more helpful to someone like Amanda the IOE would be measurement and other checks that can be take during development and that can help control what the final system will be like. The system development process in made up of a number of activities that are liked together so that the output from one activity is the input to the next step. Thus, program testing will depend on there being a program to test that will be the product of the program coding stage. Errors can enter the process at any stage. They can be introduced either because of a defect in the way a process is carried out as when programmers make mistakes in the logic of their programs or because information has not been passed clearly and unambiguously between stages.

Software Quality

Errors that creep in at the early stages are more expensive to correct at late staves for the following reasons

The later the error is found the more rework at more stages of development.

The general tendency is for each successive stage of development to be more detailed and less able to absorb change.

Error should therefore be eradicated by careful examination of the products of each stage before they are passed on to the next. The following process requirement should be specified for each activity.

Software Quality

Entry requirement. Which have to be in place before an activity can start.

Implementation requirement: Which define how the process is to be conducted.

Exit Requirement. Which have to be fulfilled before an activity is deemed to have been completed.

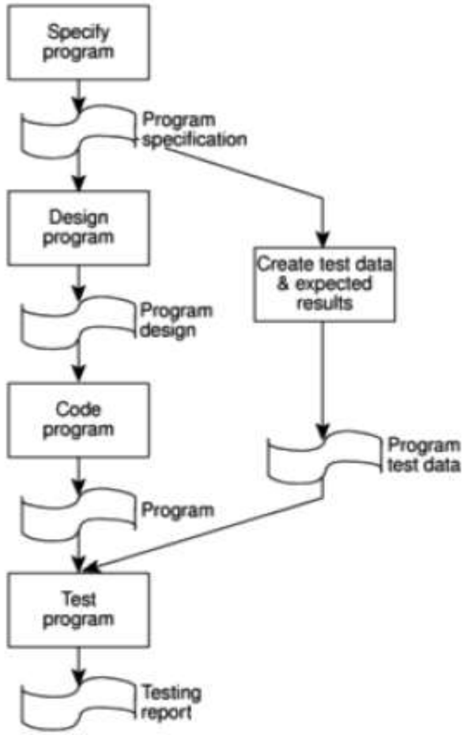


Figure 12.2 *An example of the sequence of processes and deliverables.*

Software Quality

12.8 External Standard (BS EN ISO 9001)

Various national and international standards bodies including the British Standard Institution (BSI) in United Kingdom have Inevitably become involved in the creation of standards or quality management system. The British standard is now called BS EN ISO 9001:1994. Standards such as the ISO 9000 such as the ISO 9000 series aim to ensure that a monitoring and control system to check quality is in place.

An overview of BS EN ISO QMS Requirements

In order for quality management system (QMS) to meet the standard it has to conform to certain requirement which are summarized below

- The Management must define and document the policy concerning quality and must ensure that this policy is communicated to all levels of the organization

Software Quality

- All Quality control procedures must be documented
- All contract to supply goods or services must contain mutually agreed requirements that the developers is capable for delivering.
- There must be procedures to control and verify the design of the system to be supplied so that it meets the requirements agreed with customers.
- There must be procedures to approve design and other documentation.
- Where components of the system to be supplied to the clients are obtained from third parties there must be procedures to ensure check and maintain the quality of these components.
- Individuals products must be identifiable as should their components

Software Quality

- The process by which the final product is created must be planned and monitored
- Inspection and testing must take place during the development phase at its completion and before delivery. Tests and inspection must also be carried out on the component obtained from third parties
- The equipment used in the production process itself must be properly controlled with respect to quality
- The testing status of all components and system must be clearly recorded at all times
- Care must be take to ensure that items that are known to be defective or not in carelessly used.
- When a defect is detected, measures must be undertaken to remove the defective part and to ensure that the defect does not occur again

Software Quality

- Satisfactory procedure must be in place to details with correct handling storage, packaging and delivery of the products
- Sufficient records must be maintained to demonstrate that the quality system s working satisfactorily.
- The software quality management system must be audited on a regular basis
- Servicing and support activities must be subject to the quality management system
- The developers must establish appropriate statistical techniques to verify the acceptability of the final product.

Software Quality

TickIT

The ISO9000 standard refer to quality management system is general but in the united kingdom, The department of Trade and Industry (DTI) have formulated the TICKIT standard which give an interpretation of these standards and includes the followings requirements.

- A detailed development plan is required before development is embarked upon.
- Chang control procedures should be used at all stages of development
- Design reviews must take place.
- The suitability of the design methodology must the reviewed
- Progress must be reviewed on a systematic basis.

Software Quality

- It must be possible to trace back the features of the software design to specification and requirements.
- Designs must be properly documented
- Suitable test plans, specifications and records must be produced
- A code of practice must be in place which governs the way the software is developed
- The code of practice must include the requirements that
- The design must be broken down into levels each with identifiable input and outputs
- Software must be organized into modules
- A modules must normally performs a single function or a set of related functions
- A plan language description must exist for each module

Software Quality

Capability process models

Rather than just checking that a system is in place to detect faults, a customers might wish to check that a supplier is using a software development methods and tools that are like to produce good quality software. In The United states, an influential capability maturity model (CMM) has be developed at the software engineering institute (SEI), a part of the Carnegie- Mellon University. These levels are defined as.

Level 1: Initial: The procedures followed tend to be haphazard. Some project will be successful, but this tends to be because of the skills of particular individuals including project managers. This is no level 0 and so any organization would be at this level by default.

Level 2: Repeatable: Organization at this level will have basic project management procedures in place, however, the way an individual task is carried out will largely on the person doing it

Software Quality

Level 3 Defined: the organization have the way in which each task in the software development life cycle is to be done.

Level 4 Managed: The product and the processes involved in software development are subject to measurement and control

Level 5 Optimizing: Improvement in procedure are designed and implemented using the data gather from the measurement process

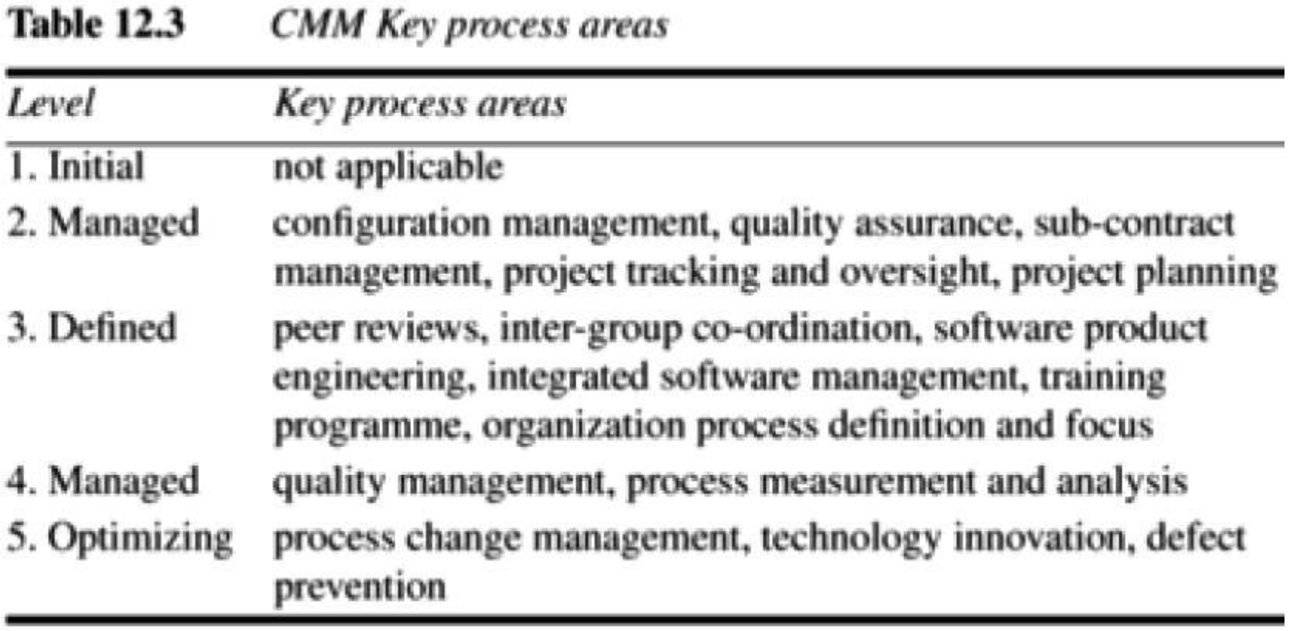


Table 12.3 *CMM Key process areas*

Level	*Key process areas*
1. Initial	not applicable
2. Managed	configuration management, quality assurance, sub-contract management, project tracking and oversight, project planning
3. Defined	peer reviews, inter-group co-ordination, software product engineering, integrated software management, training programme, organization process definition and focus
4. Managed	quality management, process measurement and analysis
5. Optimizing	process change management, technology innovation, defect prevention

Software Quality

Assessing software products:

The concern in this section has so far been with the assessment of organization and the process that they used to produce software, but many purchases of software, including project managers contemplating the purchase of software tools are more directly worried about the quality of the software product itself.

Compliers for some programming languages for examples are subject to certification. Must progress however has still to be made in this area.

Software Quality

12.9 Techniques to help Enhance software Quality

So far in this chapter we have looked at the steps a customer might take to ensure the quality of software produced by an outside supplier

Increase Visibility: A landmark in this movement towards making the software development process more visible was the advocacy by the American software guru, Gerald Weinberg of egoless programming. Weinberg encouraged the simple practice of programmer looking at each other code.

Procedural Structure: At first programmer were more or less left to get on with writing the programs, although there might be some general guidelines, Over the years there has been the growth of methodologies where every process in the software development cycle has carefully laid down steps

Software Quality

Checking intermediate stages:

It seem inherent in human nature to push forward quickly with the development of any engineered object until a working model, however imperfect has been produced that can then be debugged. One of the element of the move towards quality practices has been put emphasis on checking the correctness of work at its earlier conceptual stages.

The push towards visibility can be increase by using walkthroughs, inspection and reviews. The movement towards more procedural structure inevitably leads to discussion of structured programming techniques and to its later manifestation in the ideas of Clean-room software development

Software Quality

Inspection

When a piece of work is completed the copies of the work are distributed to co-worker to identifies the defects. The involved colleagues must the experience in the similar area such a programmer always inspect the work of programmes. Our own experience of using this techniques has been that.

- It is very effective way of removing superficial error from a piece of work
- It motivate the programmer to produce a better structure and self explanatory program because they know that other peoples will criticizing it
- It helps spread good programming practices because the participants discuss the advantages and dis- advanteges of specific pieces of code.
- It can enhance team spirit.

Software Quality

General Principal behind the Fagan method

- Inspections are carried our on all major deliverables
- All type of defects are noted
- Inspection can be carried out by colleagues
- Inspection are carried out using a predefined set of steps
- Inspection meeting do not last for more than two hours
- The inspection is led by a moderator who has had specific training in the techniques
- The participants have define rules
- Checklist are used to assist the fault-finding process
- Material is inspected at an optimal rate of about 100 lines an hour
- Statistics are maintained so that the effectiveness of the inspection process can be monitored

Software Quality

Structured Programming and clean-room software development

The way to deal with complex system, it was contended was to break them down into components that were of a size for the human mind to comprehend. For a large system there would be a hierarchy of components and sub-components. For this decomposition to work properly each components would have to be self contained with only one entry and exit point. The Idea of structure programming have been further developed into the ideas of clean room software development by the peoples such as Harlan Mills f IBM with this type of development there are three separate teams.

A specification team, which obtains the user requirements and also a usage profile estimating the volume of use for each feature in the system

Software Quality

- A Development team, which develop the code but does no machine testing of the program code produced
- A certification team which carries out testing

A system is produced in increments each of which should be capable of actual operation by the end user. The development team does no debugging. Instead all software has to be verified by them using mathematical techniques.

Software Quality

Formal Methods

In the section above on clean room development the use of mathematical verification techniques was mentioned. These techniques used un-ambiguous. Mathematically-based, specification. They are used to define pre-conditions and post conditions for each procedures. Pre-conditions define the allowable states, before processing or the various items of data that a procedures is to work upon. The post-conditions define the state of these data items after the procedure has been executed.

Software Quality Circles:

Must interest has been shown in Japanese software quality practices. The aim of the Japanese approach is to examine and modify the activities in the development process in order to reduce the number of error that they have in their end products. Testing and Fagan Inspection can assist the removal or error but the same types of error generally occur again and again in successive products created by a specific type of process. By uncovering the source of errors, this repetition can be eliminated.

Software Quality

To do this needs the involvement of all the staff in identifying the causes of errors. The staff are involved in the identification of source through the formation of quality circles known as Software Quality circle (SWQC).

- Identify a list of problems
- Select one problems to solve
- Clarify the problem
- Identify and evaluate the causes
- Identify and evaluate the solutions
- Decide one solution
- Develop and implementation plan
- Present the plan to management
- Implement the plan
- Monitor the plan
- Consider wider applicability of solution
- Restart from B

Software Quality

A number of specific techniques characterize quality circle, the most prominent of which is brainstorming. As a idea is written down the on a flip card for comments and criticisms and suggestion.

The effectiveness of Quality circles:

For quality circles to work there be full support for them at all levels of management. First-line management can feel threatened by them as they might appear to undermine their authority. After all problem solving is one of their main tasks. The proponents of quality circle see them as a way of giving management more time for planning and development. An manager will have to devote time to fire-fighting dealing with ad hoc crises and then can detract from longer term activities that will be able to improve the effectiveness of the organization.

Software Quality

The GQM Approach:

Approaches to process improvement such as the quality circles that have just discuss, May be support by quantitative measurement techniques. One of these is the GQM (Goal/Question/Metric) approach. This require that the goals to achieved is firstly identified. A goal might be to evaluate whether a new programming language allowed developers to be more productive. In order to achieve this goal certain specific question now need to be identified as needing answer. For example in order to evaluate the productivity of a new programming language the following questions might need answering.

- How quick where developers previously writing code
- How quickly are developers writing code with new programming language

Software Quality

How good the quality the software previously

How good is the quality of the software produced using the new programming language

For each question, a number of matrices will need to be identifies as needing collection in order to answer the question. For instance, to find out how quickly developers have ben writing code, development efforts and the size of development tasks in terms of the number of function point to be produced might be needed

Thank you

Department of Computer Science , National Textile University Faisalabad-37610, Pakistan
Office Phone: +92-41-9230081 Ext: 119 | Fax: +92 (41) 9230098 | email: cmn.faisal@ntu.edu.pk