Software Quality

Software Quality

1 Importance of Software Quality

Increasing criticality of software - the final customer or user is anxious about the general quality of software, especially its reliability. This is increasingly in case where organizations depend on computer systems and software for their operations

2 The Intangibility of software

It is difficult to know whether a particular task in a project has been completed satisfactorily.

3.Accumulating errors during software development

as a computer system consists of number of steps, where output from one step is input to next step, errors in one step, will be added to other steps. Later in a project, an error which is found will be more expensive to fix.

Defining Software Quality

For any software system there should be three specifications

1. A **functional specification** describing what the system is to do
2. A **quality specification** concerned with how well the functions are to operate
3. A **Resource specification** concerned with how much is to be spent on the system

Software Quality factors

1. Product operation qualities
2. Product revision qualities
3. Product transition qualities

1. Product operation quality factors

*** Correctness -** extent to which a program satisfies its specifications and fulfils the user's objectives

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

*** Efficiency -** the amounts of computer resources required by the software

*** Integrity -** extent to which access to software or data by unauthorized persons can be controlled

* **Usability -** effort required to learn, operate, prepare input and interpret output

2. Product revision quality factors

1. Maintainability - effort required to locate and fix an error in an operational program

2. Testability - effort required to test a program to ensure it performs its intended function

3. Flexibility - effort required to modify an operational program

3. Product Transition Quality factors

1. Portability - effort required to transfer a program from one hardware configuration software system environment to another

2. Reusability - extent to which a program can be used in other applications

3. Interoperability- effort required to link one system to another

James A. Mccall's 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, I/O volume
Maintainability	simplicity, self-descriptiveness, consistency
Testability	simplicity, modularity, self descriptiveness
Flexibility	generality, expandibility, self descriptiveness
Portability	machine independence, software system independence
Reusability	generality, software system independence, machine independence, self-descriptiveness
Interoperability	communications commonality, data commonality

Class Exercise :

Divide the class into 3 groups
Each group should select a team leader

Each group should identify a pair of quality attributes discussed in the class for 3 different types of categories

1) Complementary : The Presence of this Attribute will improve the performance of other attribute.
2) Conflicting : The presence of this attribute will affect the other attribute
3) Indifferent : The presence of this attribute has no effect on the other attribute.

How to measure the quality ?

- The following should be laid down for each quality

1 Scale - the unit of measurement

2 Test - the practical test of the extent to which the attribute quality exists

3. Worst - the worst acceptable value

4. Plan - the value that is planned to achieve

5. Best - the best value that appears to be feasible; this would be a level that is known to have been achieved elsewhere

6. Now - the value that applies currently

Eg. Suggest quality specifications for a word processing package

Quality - Ease of learning

definition - the time taken, by a novice, to learn how to operate the package to produce a standard document

scale - hours

test - interview novices to ascertain their previous experiences of word processing, supply them with a machine, software a training manual and a standard document to type. Time how long it takes them to learn how to type a document

worst - 4 hours

planned - 2 hours

best - 1 hour

now - 4 hours

Practical Software Measures

The measures described relate to the final software products of a project.

Reliability

This might be measured in terms of:

* Availability - the percentage of a particular time interval that a system is usable
* MTBF - the total service time divided by number of failures
* failure on demand - the probability that a system will not be available at the time required or the probability that a transaction will fail
* support activity - the number of fault reports that are generated

Maintainability

This is closely related to flexibility, the ease with which the software can be modified. The main difference is that before an amendment can be made, the fault has to be diagnosed first.

Diagnosability-avg amout of time needed to diagnose a fault

Extendibility

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.

Product versus Process quality Management

* Measurements described above can be done only after system is operational
* If measurements & other checks are taken during development that could help us a lot.
* System development process is made up of number of activities that are linked together so that output from one activity is input to the next activity. So program testing should be done at each stage.
* Errors should be corrected by careful examination of the deliverables of each stage before they are passed on to the next.

To do this the following **process requirements** should be specified for each activity.

Railway Reservation system is available from 8am to 8pm on all days. Over a four week period the system was not available on two days. On a day the system was down due to network Problems from 8am to 10am. On another day the system was not available from 10am to 2pm due to virus problems. Compute the MTBF (Mean Time Between Failures) and the availability of the given service.

Each day system should be available from 8am to 8pm.
i.e. 12 hours
Over 4 weeks that should be 12 x 7 x 4 hours=336 hours

It was unavailable for one day for 2 hours
On another day it was unavailable for 4 hours

Total hours available=336-2-4=330 hours

Availability 330/336 x 100 = 98.2%

MTBF = 330/2 = 165 hours

The Original maintenance billing system contains 5000 LOC (Source Lines of Code) and took 400 work days to implement. An amendment to the core system caused by the introduction of group accounts has led to 100 SLOC being added which took another 20 workdays to implement.

1) Calculate the productivity of original system
2) Calculate the productivity of the amendment
3) Calculate the Extendibility

Productivity of the original system
5000/400 = 12.5 SLOC/Staff-day

Productivity of the amendment
100/20 = 5 SLOC/Staff-day

Extendibility =
5/12.5 X 100 = 40%

Techniques to help enhance software quality

Three main themes emerge :

1. Increasing visibility- Software development process should be made more visible by the simple practice of programmers looking at each other's code - "egoless programming" by Gerald Weinberg

2. Procedural structure - Standard procedure must be followed for software development cycle

3. Checking intermediate stages- We should adopt the quality practice of checking the correctness of work at its earlier stage itself

Structured programming and clean-room software development

E.W. Dijkstra (digstra) in 1968, wrote a letter to a computer journal which was entitled “Go To statement is considered harmful” which led to the common idea that structured programming was simply not using GO TO statements

The ideas of structured programming have been further developed into the ideas of clean-room software development by **Harlan Mills** of IBM, who developed this idea.

With this development, there are three separate teams

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

* **a development team-** which develops the code but which does no machine testing of the program code produced

* **a certification team -** which carries out testing