Software Engineering
College of Arts, Media and Technology ,CMU.

SE422 Software Quality Assurance
CH-3 Software Quality Factor

Kittitouch S.
1.2-19-11-12

Topics

- Quality Factor
- The need for comprehensive software quality requirements
- Classifications of software requirements into software quality factors
  - McCall's factor model

Case Study One

"Our new sales information system seems okay, the invoices are correct, the inventory records are correct, the discounts granted to our clients exactly follow our very complicated discount policy, **but** our new sales information system frequently fails, usually at least twice a day, each time for twenty minutes or more. Yesterday it took an hour and half before we could get back to work . . . . Imagine how embarrassing it is to store managers . . . . Softbest, the software house that developed our computerized sales system, claims no responsibility . . . ."

Case Study Two

"Just half a year ago we launched our new product – the radar detector. The firmware RD-8.1, embedded in this product, seems to be the cause for its success. But, when we began planning the development of a European version of the product, we found out that though the products will be almost similar, our software development department needs to develop new firmware; almost all the design and programming will be new."

Case Study Three

"Believe it or not, our software package 'Blackboard' for schoolteachers, launched just three months ago, is already installed in 187 schools. The development team just returned from a week in Hawaii, their vacation bonus. But we have been suddenly receiving daily complaints from the 'Blackboard' maintenance team. They claim that the lack of failure detection features in the software, in addition to the poor programmer's manual, have caused them to invest more than the time estimated to deal with bugs or adding minor software changes that were agreed as part of purchasing contracts with clients."

Case Study Four

"The new version of our loan contract software is really accurate. We have already processed 1200 customer requests, and checked each of the output contracts. There were no errors. **But** we did face a severe unexpected problem – training a new staff member to use this software takes about two weeks. This is a real problem in customers' departments suffering from high employee turnover . . . . The project team says that as they were not required to deal with training issues in time, an additional two to three months of work will be required to solve the problem."

There are some characteristic common

- All the software projects satisfactorily fulfilled the basic requirements for correct calculations (correct inventory figures, correct average class's score, correct loan interest, etc.).
- All the software projects suffered from poor performance in important areas such as:
  - First have problem in reliability.
  - Second have problem in reusability.
  - Third have problem in maintainability.
  - Fourth have problem in Training.

Why?

There is a need for a comprehensive definition of requirements that will cover all attributes of software and aspects of the use of software, including

- Usability aspects.
- Reusability aspects.
- Maintainability aspects.
- In order to assure the full satisfaction of the users.

Quality Factors

The great variety of issues related to the various attributes of software and its use and maintenance, as defined in software requirements documents, can be classified into content groups called ***quality factors***.

Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product operation factors**:

    Correctness, Reliability, Efficiency, Integrity, Usability.

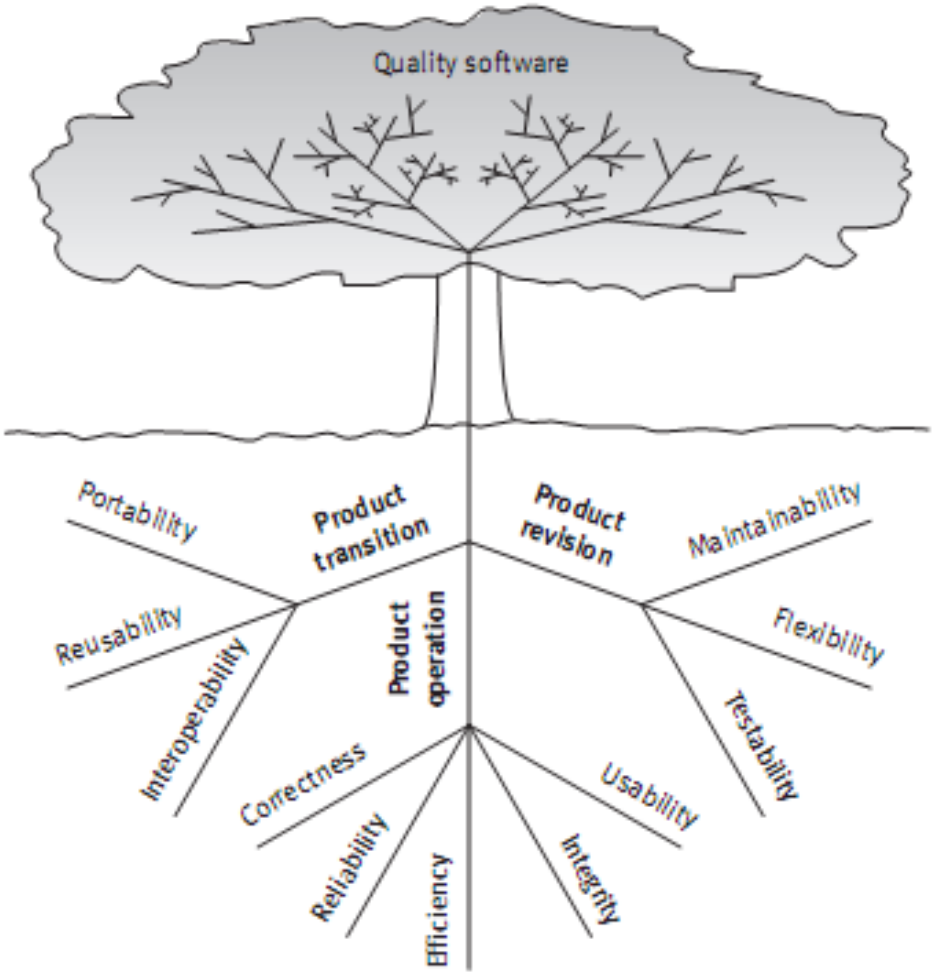
  - **Product revision factors**:

    Maintainability, Flexibility, Testability.

  - **Product transition factors**:

    Portability, Reusability, Interoperability.

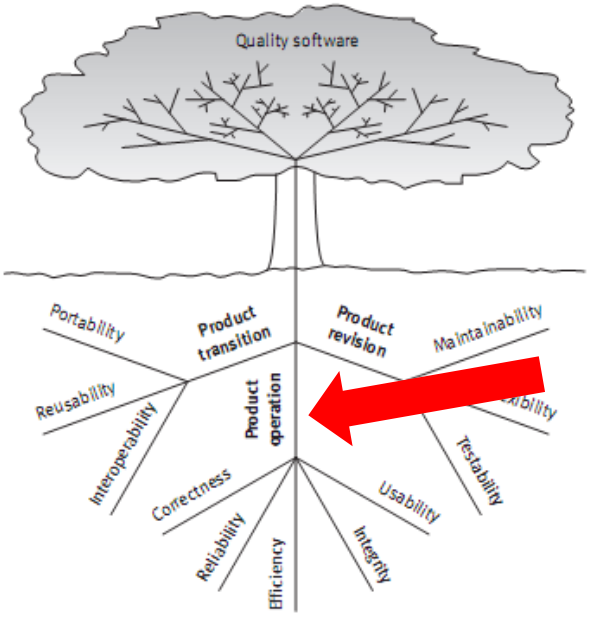
Figure 3.1: McCall's factor model tree



Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product operation factors**:

    Correctness, Reliability, Efficiency, Integrity, Usability.



Reliability

- Reliability requirements *deal with failures* to provide service. They determine the maximum allowed software system failure rate, and can refer to the entire system or to one or more of its separate functions.

- Sub-Factors:
  - Application reliability.
  - Hardware failure recovery.

Examples on Reliability

- The failure frequency of a heart-monitoring unit that will operate in a hospital's intensive care ward is required to be less than one in 20 years. Its heart attack detection function is required to have a failure rate of less than one per million cases.

- One requirement of the new software system to be installed in the main branch of Independence Bank, which operates 120 branches, is that it will not fail, on average, more than 10 minutes per month during the bank's office hours. In addition, the probability that the off-time (the time needed for repair and recovery of all the bank's services) be more than 30 minutes is required to be less than 0.5%.

Correctness

- Correctness requirements are defined in a list of the software system's required outputs.
  - Output mission.
  - Accuracy.
  - Product operation Completeness.
  - Up-to-date.
  - Availability (response time).
  - Coding and documentation guidelines.

Examples on Correctness

- **Mission**: A defined list of 11 types of reports, four types of standard letters to members and eight types of queries, which were to be displayed on the monitor on request.
- **Completeness**: The probability for a non-accurate output, containing one or more mistakes, will not exceed 1%.
- **Accuracy**: The probability of missing data about a member, his attendance at club events, and his payments will not exceed 1%.

Examples on Correctness

- **Up-to-date**: Not more than two working days for information about participation in events and not more than one working day for information about entry of member payments and personal data.
- **Availability**: Reaction time for queries will be less than two seconds on average; the reaction time for reports will be less than four hours.
- **The required standards and guidelines**: The software and its documentation are required to comply with the client's guidelines.

Integrity

- Integrity requirements deal with the software system security, that is, requirements to prevent access to unauthorized persons, to distinguish between the majority of personnel allowed to see the information ("read permit") and a limited group who will be allowed to add and change data ("write permit"), and so forth.

- Sub-Factors:
  - Access control.
  - Access audit.

Examples on Integrity

The Engineering Department of a local municipality operates a GIS (Geographic Information System). The Department is planning to allow citizens access to its GIS files through the Internet. The software requirements include the possibility of viewing and copying but not inserting changes in the maps of their assets as well as any other asset in the municipality's area ("read only" permit). Access will be denied to plans in progress and to those maps defined by the Department's head as limited access documents.

Efficiency

- **Efficiency requirements** deal with the hardware resources needed to perform all the functions of the software system in conformance to all other requirements. The main hardware resources to be considered are the computer's processing capabilities (measured in MIPS – million instructions per second ,etc.)

- Sub-Factors:
  - Efficiency of processing.
  - Efficiency of storage.
  - Efficiency of communication.
  - Efficiency of power usage (for portable units).

Examples on Efficiency

A chain of stores is considering two alternative bids for a software system. Both bids consist of placing the same computers in the chain's headquarters and its branches. The bids differ solely in the storage volume: 20 GB per branch computer and 100 GB in the head office computer (Bid A); 10 GB per branch computer and 30 GB in the head office computer (Bid B). There is also a difference in the number of communication lines required: Bid A consists of three communication lines of 28.8 KBPS between each branch and the head office, whereas Bid B is based on two communication lines of the same capacity between each branch and the head office. In this case, it is clear that Bid B is more efficient than Bid A because fewer hardware resources are required.

Usability

- Usability requirements *deal with the scope of staff resources* needed to train a new employee and to operate the software system.

- Sub-Factors:
  - Operation Resources.
  - Training.

Examples on Usability

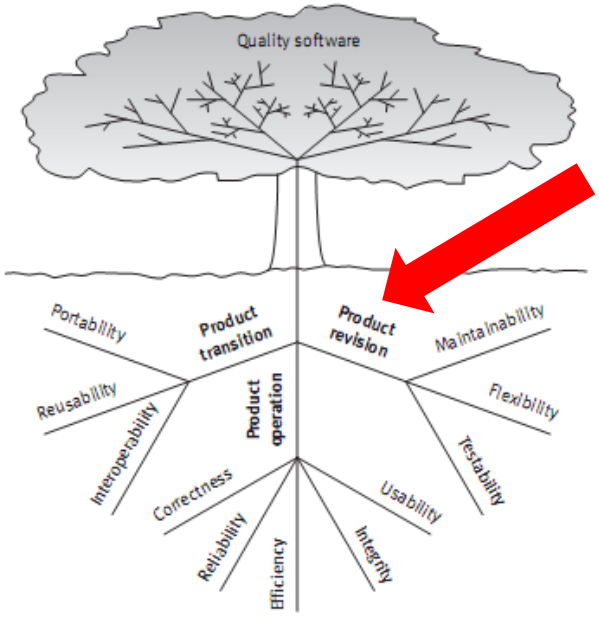
The software usability requirements document for the new help desk system initiated by a home appliance service company lists the following specifications:

- A staff member should be able to handle at least 60 service calls a day.
- Training a new employee will take no more than two days (16 training hours), immediately at the end of which the trainee will be able to handle 45 service calls a day.

Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product revision factors**:

    Maintainability, Flexibility, Testability.



Maintainability

- Maintainability requirements determine 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.

- Sub-Factors:
  - Simplicity.
  - Product revision Modularity.
  - category Self-descriptiveness.
  - Coding and documentation guidelines.
  - compliance (consistency).
  - Document accessibility.

Examples on Maintainability

- The size of a software module will not exceed 30 statements.
- The programming will adhere to the company coding standards and guidelines.

Flexibility

- The capabilities and efforts required to support adaptive maintenance activities are covered by the flexibility requirements. These include the resources(i.e. in man-days) required to adapt a software package to a variety of customers of the same trade, of various extents of activities, of different ranges of products and so on.

- Sub-Factors:
  - Modularity.
  - Generality.
  - Simplicity.
  - Self-descriptiveness.

Examples on Flexability

TSS (teacher support software) deals with the documentation of pupil achievements, the calculation of final grades, the printing of term grade documents, and the automatic printing of warning letters to parents of failing pupils. The software specifications included the following flexibility requirements:

- The software should be suitable for teachers of all subjects and all school levels (elementary, junior and high schools).
- Non-professionals should be able to create new types of reports according to the schoolteacher's requirements and/or the city's education department demands.

Testability

- Testability requirements deal with the testing of an information system as well as with its operation. Testability requirements for the ease of testing are related to special features in the programs that help the tester, for instance by providing predefined intermediate results and log files.
- Sub-Factors:
  - User testability.
  - Failure maintenance testability.
  - Traceability.

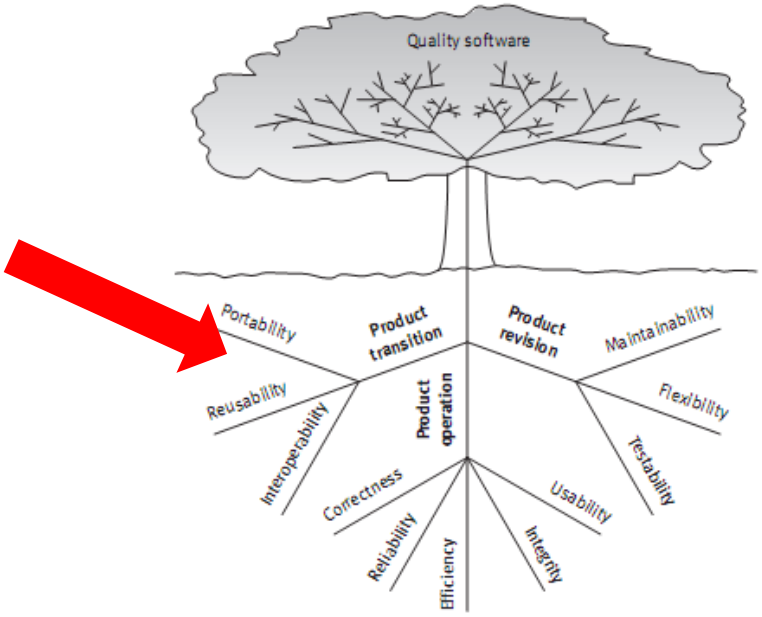
Examples on Testability

An industrial computerized control unit is programmed to calculate various measures of production status, report the performance level of the machinery, and operate a warning signal in predefined situations. One testability requirement demanded was to develop a set of standard test data with known system expected correct reactions in each stage. This standard test data is to be run every morning, before production begins, to check whether the computerized unit reacts properly.

Classifications of software requirements into software quality factors

- McCall's factor model
  - **Product transition factors**:

    Portability, Reusability, Interoperability.



Portability

- **Portability requirements** tend to the adaptation of a software system to other environments consisting of different hardware, different operating systems.
- Sub-Factors:
  - Software system independence.
  - Modularity.
  - Self descriptive.

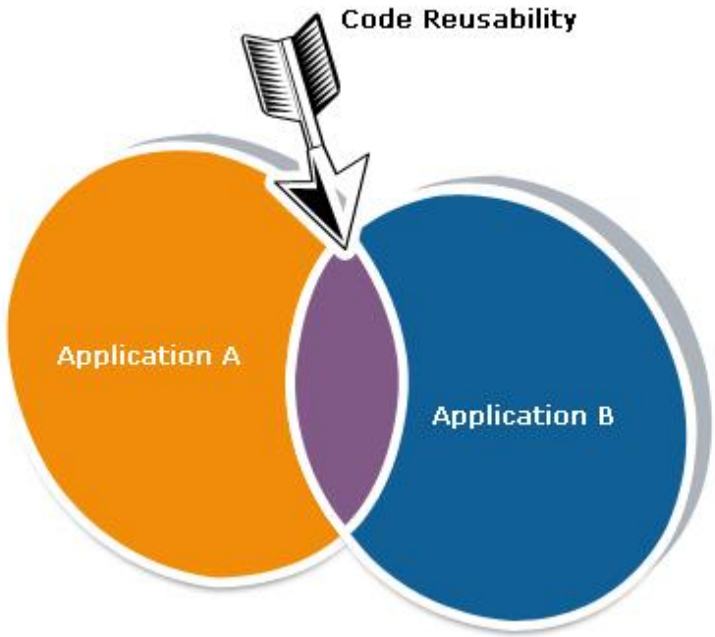
Examples on Portability

A software package designed and programmed to operate in a Windows 2000 environment is required to allow low-cost transfer to Linux and Windows NT environments.

Reusability

- Reusability requirements deal with the use of software modules originally designed for one project in a new software project currently being developed.

- Sub-factors:
  - Modularity.
  - Document accessibility.
  - Software system independence
  - Application independence.
  - Self descriptive.
  - Generality.
  - Simplicity.



Examples on Reusability

A software development unit has been required to develop a software system for the operation and control of a hotel swimming pool that serves hotel guests and members of a pool club. Although the management did not define any reusability requirements, the unit's team leader, after analyzing the information processing requirements of the hotel's spa, decided to add the reusability requirement that some of the software modules for the pool should be designed and programmed in a way that will allow its reuse in the spa's future software system, which is planned to be developed next year. These modules will allow:

- Entrance validity checks of membership cards and visit recording.
- Restaurant billing.
- Processing of membership renewal letters.

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).
- Interoperability requirements can specify the name(s) of the software or firmware for which interface is required. They can also specify the output structure accepted as standard in a specific industry or applications area.

Examples on Interoperability

The firmware of a medical laboratory's equipment is required to process its results (output) according to a standard data structure that can then serve as input for a number of standard laboratory information systems.

Table 3.1: Comparison of McCall's factor model and alternative models

No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciniak	Deutsch and Willis
1	Correctness	+	+	+
2	Reliability	+	+	+
3	Efficiency	+	+	+
4	Integrity	+	+	+
5	Usability	+	+	+
6	Maintainability	+	+	+
7	Flexibility	+	+	+
8	Testability	+		
9	Portability	+	+	+
10	Reusability	+	+	+
11	Interoperability	+	+	+
12	Verifiability		+	+
13	Expandability		+	+
14	Safety			+
15	Manageability			+
16	Survivability			+

Factors of the alternative models

- **Verifiability**: define design and programming features that enable efficient verification of the design and programming.
- **Expandability**: future efforts that will be needed to serve larger populations, improve service, or add new applications in order to improve usability.
- **Safety**: meant to eliminate conditions hazardous to operators of equipment as a result of errors in process control software. These errors can result in inappropriate reactions to dangerous situations or to the failure to provide alarm signals when the dangerous conditions to be detected by the software arise.

Factors of the alternative models

- **Manageability:** Manageability requirements refer to the administrative tools that support software modification during the software development and maintenance periods, such as configuration management, software change procedures, and the like.

Who is interested in the definition of quality requirements?

- Some quality factors not included in the typical client's requirements document may, in many cases, interest the developer. The following list of quality factors usually interest the developer whereas they may raise very little interest on the part of the client:
  - Portability.
  - Reusability.
  - Verifiability.

Software compliance with quality factors

- quality factors is examined by design reviews, software inspections, software tests, and so forth. Comprehensive discussions of design reviews, software testing, software quality metrics and other tools for verifying and validating the quality of software are provided in the balance of this book.

References

- Chapter 3: Daniel Galin. SOFTWARE QUALITY ASSURANCE From theory to implementation. Pearson Education Limited, 2004.