SOFTWARE QUALITY Assurance & Test

LECTURE # 3

SOFTWARE QUALITY-II

chenbo@etao.net

Last Lecture

- **Quality**
- **Quality Types**
- **Quality Control**
- **Quality Engineering**
- **Different views of Quality**
- **Quality Models**

Topics to Cover

- Quality Models
  - ISO 9126
  - FURPS
  - Boehm's
  - GQM
- Cost of Quality
  - Prevention Cost
  - Appraisal Cost
  - Failure Cost
- Quality Cost Conformance Model

Quality Models

- A **Quality Model** is defined as, “ the set of characteristics and the relationships between them which provides the basis for specifying quality requirements and evaluating quality”

- Software quality is described by specific quality models

- **Standard quality models**
  - **McCall (11 quality factors, 23 quality criteria)**
  - **ISO/IEC 9126 (6 characteristics, 27 sub-characteristics)**
  - **Boehm‟s Model (7 quality factors, 15 quality criteria)**

- **Application or company specific quality models**
  - **FURPS**
  - **GQM Approach**

McCall’s Factor-Criteria-Metrics Model

Classification into :

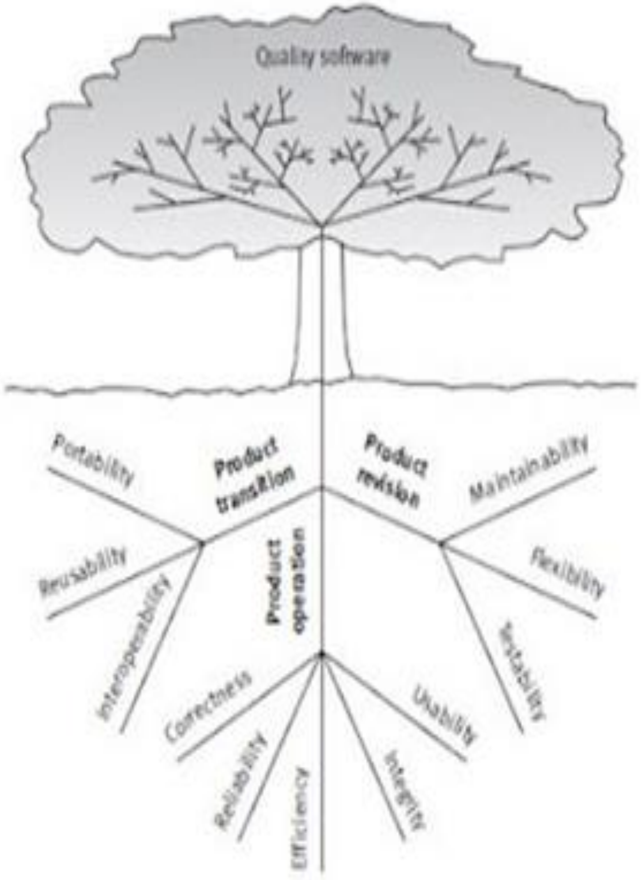
- **Factors (to specify):**
  - They describe the external view of the software, as viewed by the users.
- **Criteria (to build):**
  - They describe the internal view of the software, as seen by the developer.
- **Metrics (to control):**
  - They are defined and used to provide a scale and method for measurement.

McCall’s Quality Factors and Criteria

- McCall, Richards, and Walters studied the concept of software quality in terms of two key concepts as follows:
  - *quality factors*, and
  - *quality criteria*.
- A **quality factor** represents the behavioral characteristic of a system.
  - Examples: correctness, reliability, efficiency, testability, portability, maintainability etc.
- A **quality criterion** is an attribute of a quality factor that is related to software development.
  - Example: Modularity is an attribute of the architecture of a software system. A highly modular software allows designers to put cohesive components in one module, thereby increasing the maintainability of the system.

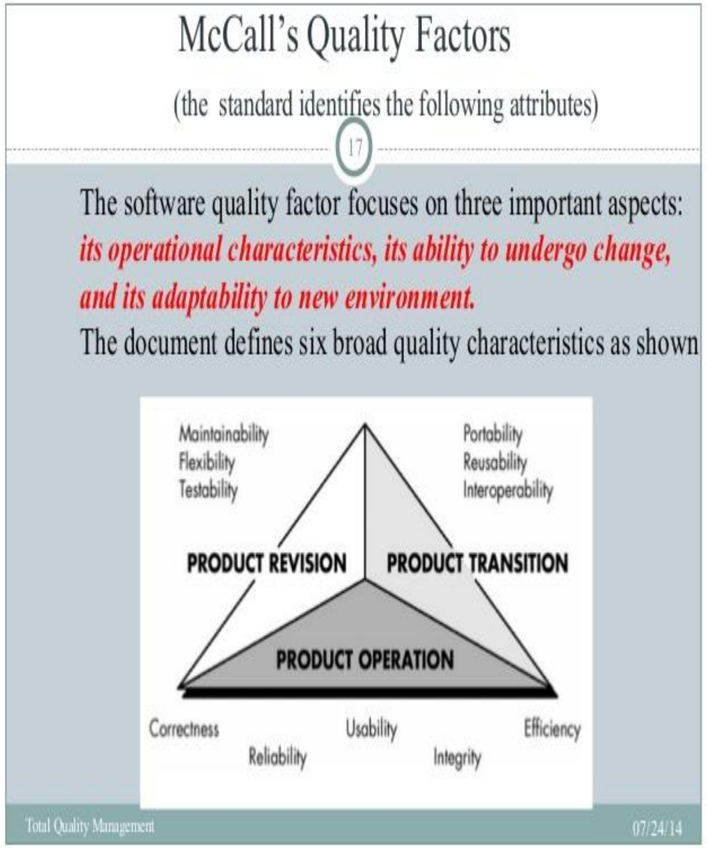
McCall’s Factor Model Tree

- The McCall Model aimed at system developers
- It is used during development process
- It identifies 3 areas of software work:
    - Product Revision (ability to undergo changes)
    - Product Transition (adaptability to new environments)
    - Product Operation (its operation characteristics)
- Doesn’t say much about Functionality



McCall’s 11 Quality Factors

- **Product Revision**
  - **Maintainability** - *Can I fix it?*
  - **Flexibility** - *Can I change it?*
  - **Testability** - *Can I test it?*
- **Product Transition**
  - **Portability** - *Will I be able to use on another machine?*
  - **Reusability** - *Will I be able to reuse some of the software?*
  - **Interoperability** - *Will I be able to interface it with another application?*
- **Product Operation**
  - **Correctness** - *Does it do what I want?*
  - **Reliability** - *Does it do it accurately all the time?*
  - **Efficiency** - *Will it run on my machine as well as it can?*
  - **Integrity** - *Is it secure?*
  - **Usability** - *Can I run it?*



McCall’s Quality Factors

(the standard identifies the following attributes)

17

The software quality factor focuses on three important aspects: *its operational characteristics, its ability to undergo change, and its adaptability to new environment.*

The document defines six broad quality characteristics as shown

Total Quality Management

07/24/14

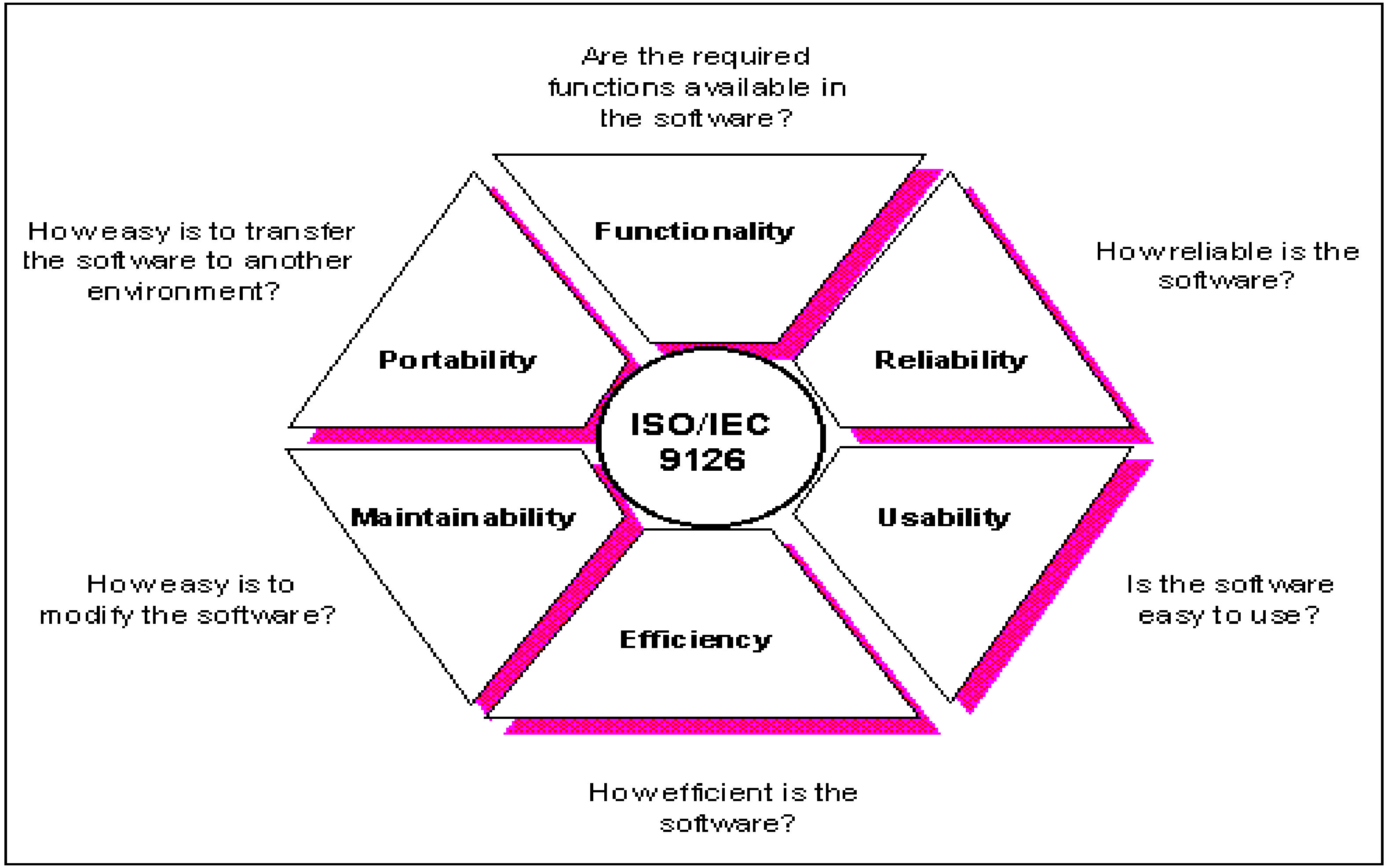
McCall’s Quality Criteria

- **Some Quality Criteria‟s from McCall‟s Model is listed below:**
  - **Accuracy** (The precisions of computations and outputs)
  - **Completeness** (The degree to which the full implementation of the required functionalities has been achieved)
  - **Error Tolerance** (The degree to which the continuity of operations is ensured under adverse conditions)
  - **Expandability** (The degree to which software functions can be expanded)
  - **Hardware Independence** (The degree to which the software is dependent on the underlying hardware)
  - **Modularity** (The provision of highly independent modules)
  - **Simplicity** (The ease with which the software can be understood)

McCall’s Quality Factors and Criteria

- **Relationship Between Quality Factors and Quality Criteria**
  - Each quality factor is positively influenced by a set of quality criteria, and the same quality criterion impacts a number of quality factors.
    - **Example: Simplicity impacts reliability, usability, and testability.**
  - If an effort is made to improve one quality factor, another quality factor may be degraded.
    - **Example: Portable code may be less efficient.**
  - Some quality factors positively impact others.
    - **Example: An effort to improve the correctness of a system will increase its reliability.**

ISO- 9126

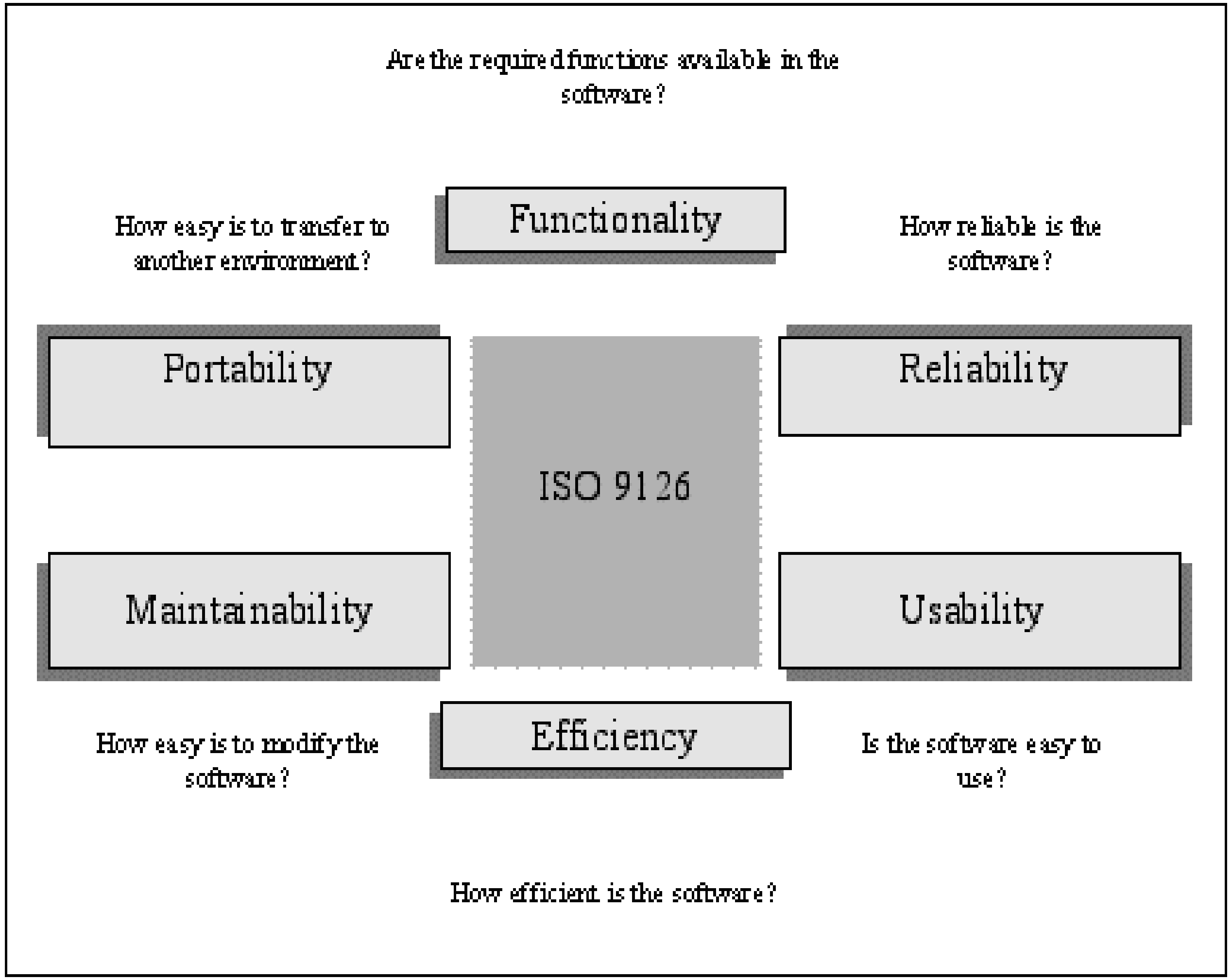


Are the required functions available in the software?
Functionality
How easy is to transfer the software to another environment?
Portability
How reliable is the software?
Reliability
ISO/IEC 9126
Maintainability
Usability
How easy is to modify the software?
Efficiency
Is the software easy to use?
How efficient is the software?

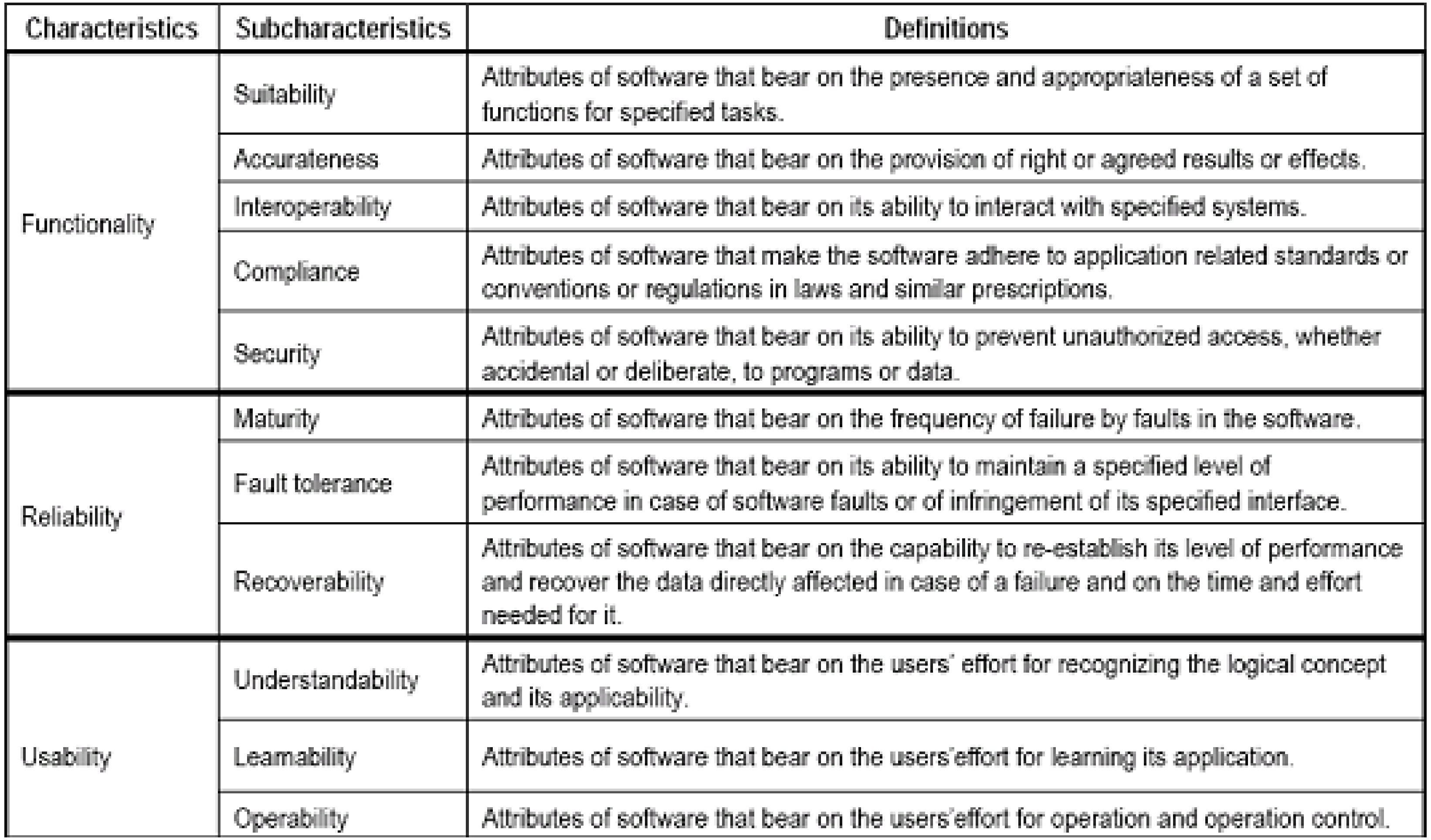
ISO-9126 Model

- **Software quality characteristics:**
  - *A set of attributes of a software product by which its quality is described and evaluated.*
  - *A software quality characteristic may be refined into multiple levels of sub characteristics. (ISO 9126:*1991 ,3.13)
  - Each sub-characteristic is evaluated by a set of metrics.

The Six Quality Characteristics (ISO/IEC 9126)

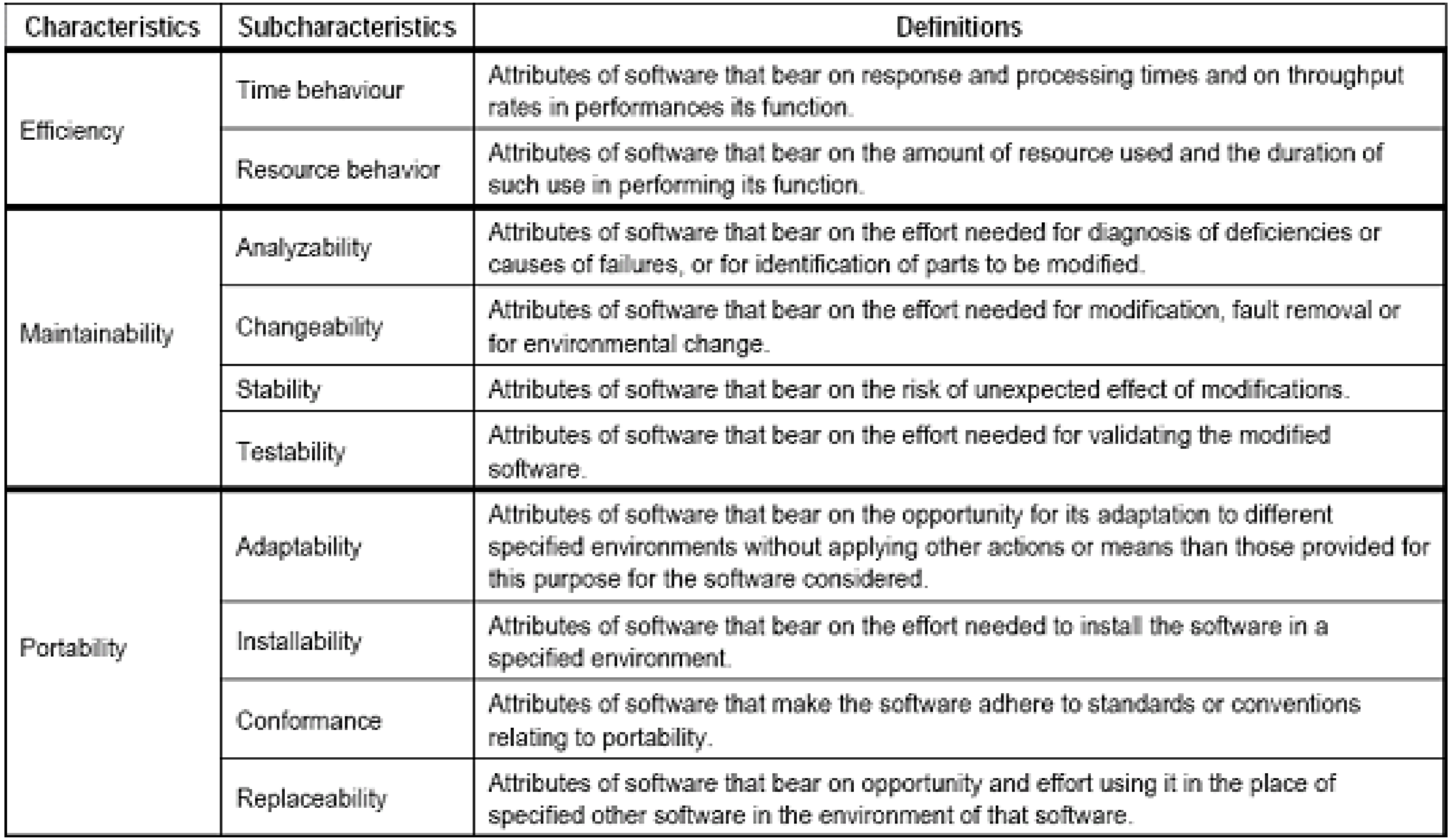

Are the required functions available in the software?
How easy is to transfer to another environment?
Functionality
How reliable is the software?
Portability
ISO 9126
Reliability
Maintainability
Usability
How easy is to modify the software?
Efficiency
Is the software easy to use?
How efficient is the software?

The Six Quality Characteristics (ISO/IEC 9126)



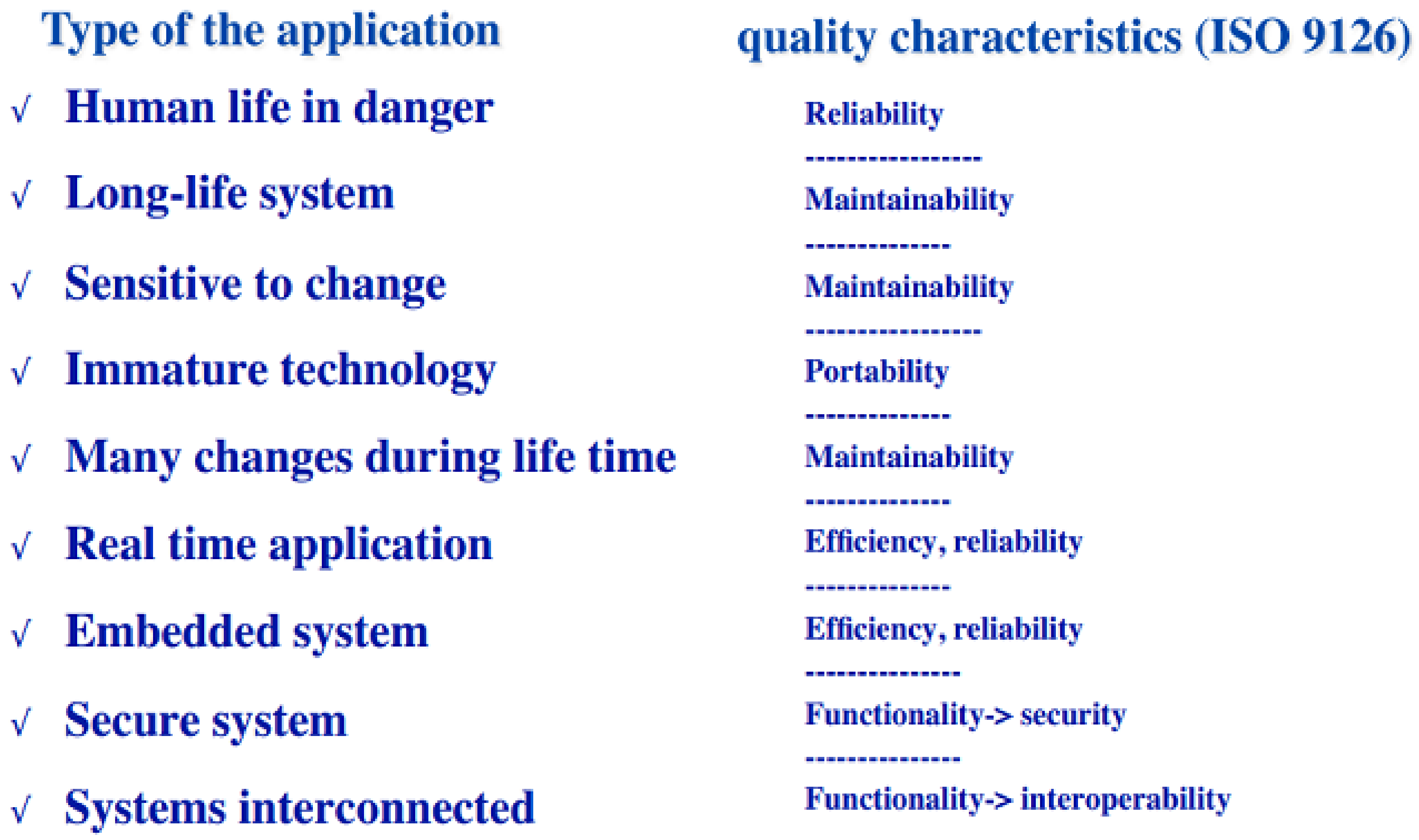
Characteristics	Subcharacteristics	Definitions
Functionality	Suitability	Attributes of software that bear on the presence and appropriateness of a set of functions for specified tasks.
	Accurateness	Attributes of software that bear on the provision of right or agreed results or effects.
	Interoperability	Attributes of software that bear on its ability to interact with specified systems.
	Compliance	Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.
	Security	Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs or data.
Reliability	Maturity	Attributes of software that bear on the frequency of failure by faults in the software.
	Fault tolerance	Attributes of software that bear on its ability to maintain a specified level of performance in case of software faults or of infringement of its specified interface.
	Recoverability	Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.
Usability	Understandability	Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability.
	Learnability	Attributes of software that bear on the users'effort for learning its application.
	Operability	Attributes of software that bear on the users'effort for operation and operation control.

The Six Quality Characteristics (ISO/IEC 9126)



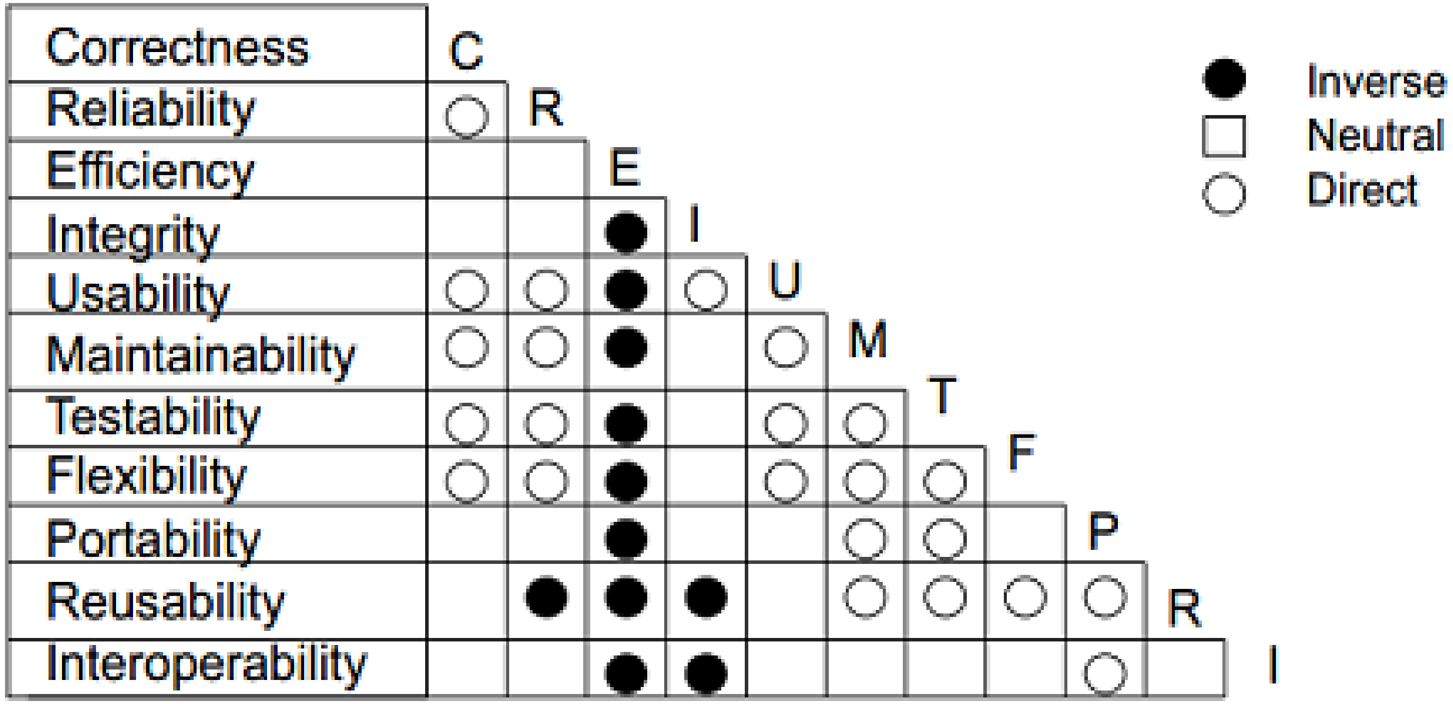
Characteristics	Subcharacteristics	Definitions
Efficiency	Time behaviour	Attributes of software that bear on response and processing times and on throughput rates in performances its function.
	Resource behavior	Attributes of software that bear on the amount of resource used and the duration of such use in performing its function.
Maintainability	Analyzability	Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.
	Changeability	Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.
	Stability	Attributes of software that bear on the risk of unexpected effect of modifications.
	Testability	Attributes of software that bear on the effort needed for validating the modified software.
Portability	Adaptability	Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.
	Installability	Attributes of software that bear on the effort needed to install the software in a specified environment.
	Conformance	Attributes of software that make the software adhere to standards or conventions relating to portability.
	Replaceability	Attributes of software that bear on opportunity and effort using it in the place of specified other software in the environment of that software.

Selecting and Prioritizing Quality Factors



Type of the application	quality characteristics (ISO 9126)
√ Human life in danger	Reliability
√ Long-life system	Maintainability
√ Sensitive to change	Maintainability
√ Immature technology	Portability
√ Many changes during life time	Maintainability
√ Real time application	Efficiency, reliability
√ Embedded system	Efficiency, reliability
√ Secure system	Functionality-> security
√ Systems interconnected	Functionality-> interoperability

Relationships between quality factors



	C	R	E	I	U	M	T	F	P	R	I
Correctness											
Reliability	○										
Efficiency											
Integrity			●								
Usability	○	○	●	○							
Maintainability	○	○	●		○						
Testability	○	○	●		○	○					
Flexibility	○	○	●		○	○	○				
Portability			●			○	○				
Reusability		●	●	●		○	○	○	○		
Interoperability			●	●					○		

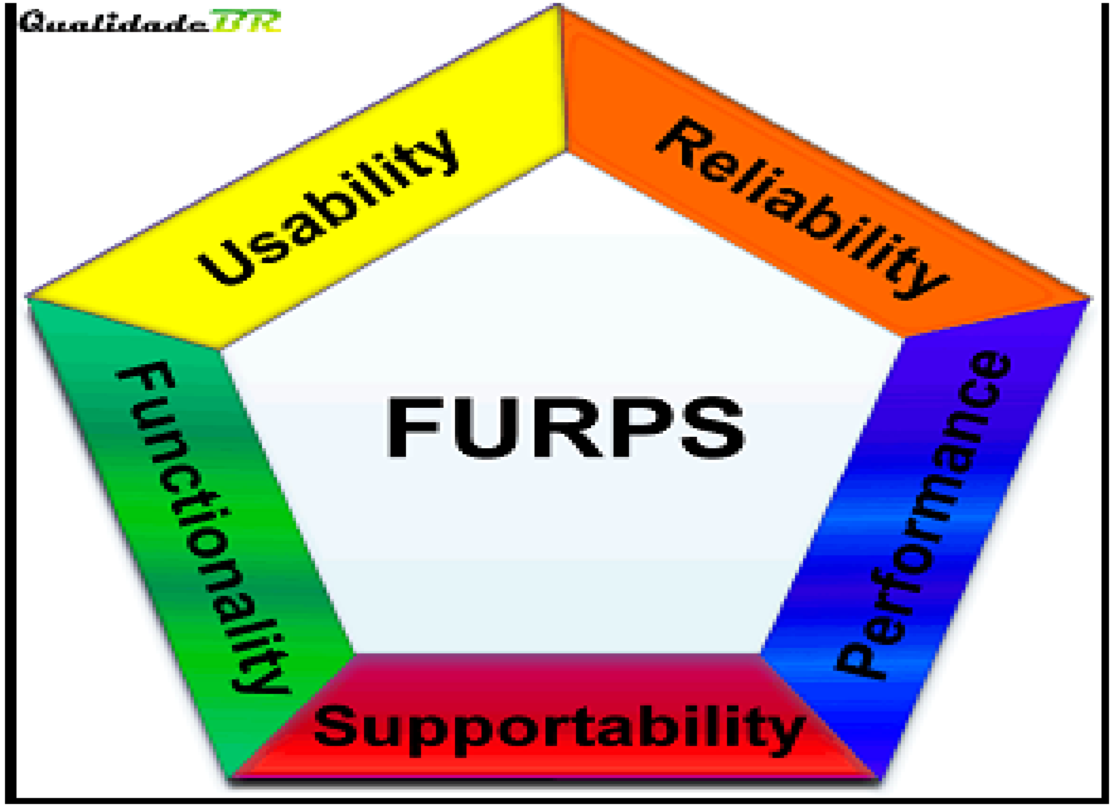
● Inverse
□ Neutral
○ Direct

Relationships between quality factors

Examples of the relationships between quality factors

- Integrity vs. efficiency (inverse)
  - The control access to data or software requires additional code and processing leading to a longer runtime and additional storage requirements.
- Usability vs. efficiency (inverse)
  - Improvements in the human/computer interface may significantly increase the amount of code required.
- Maintainability vs. flexibility (direct)
  - Maintainable code arises from code that is well structured. This will also assist any modifications or alterations that are required. Thus a direct relationship exists between these properties.

Hewlett Packard: F.U.R.P.S



Hewlett Packard: F.U.R.P.S

- Robert Grady at Hewlett Packard proposed a model called as FURPS Model in 1987 and later extended by IBM as FURPS+
- Factors:
  - **Functionality:** represents all the system-wide functional requirements that we would expect to see described
  - **Usability:** aesthetics, consistency, documentation
  - **Reliability:** Availability, Accuracy, Recoverability
  - **Performance:** response time, resource consumption, start up time, recovery time
  - **Supportability:** can it be extended, adapted, corrected?
- FURPS is originally a company specific quality model
- *FURPS+* is now widely used in the software industry.
  - **Design Constraints**
  - **Implementation Constraints**
  - **Interface Requirements**
  - **Physical Requirements**

- The + was later added to the model after various campaigns at HP to extend the acronym to emphasize various attributes.

Hewlett Packard F.U.R.P.S+

- Design Requirements
  - E.g. a relational database is required
- Implementation requirement
  - Constrains the coding or construction e.g. required standards, platform or implementation language
- Interface requirement
  - A requirement to interact with an external item
- Physical requirement
  - A physical constraint imposed on the hardware; for example, shape, size and weight

Hewlett Packard F.U.R.P.S+

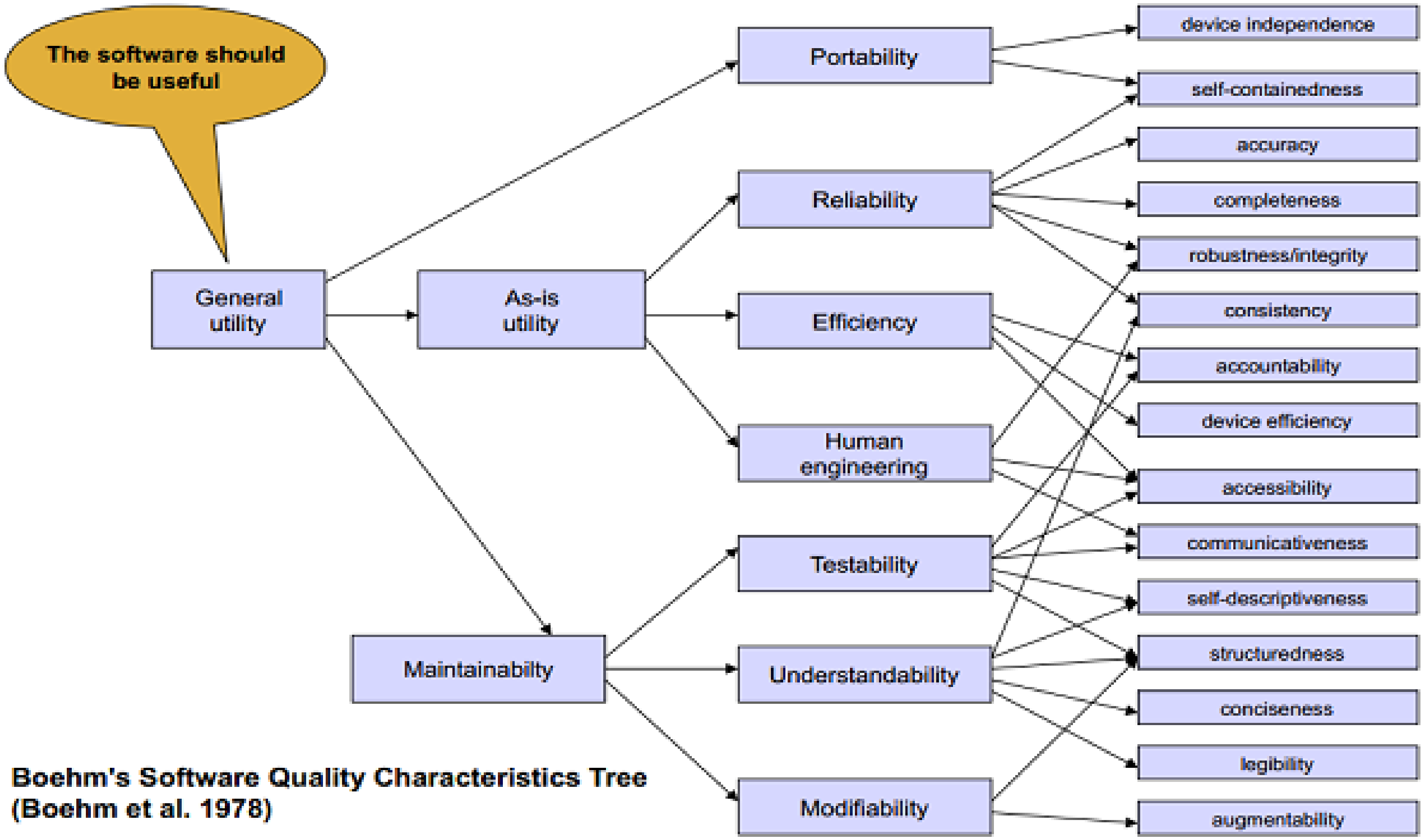
Classifying Requirements?

- The persistence will be handled by a relational database is a ------ requirement
- The database will be Oracle 8i is -------------- requirement
- The system will run 7 days a week, 24 hours a day is a ----------- requirement
- An online help system is required is a ----------- requirement
- All presentation logic will be written in Visual Basic is --------- requirement

Boehm's Model

- Another software quality model called as Boehm's quality model was given by Barry W. Boehm in 1978.
- **7 quality factors, 15 quality criteria**
- At the highest level of his model, Boehm defined three primary uses namely, as-is utility, the extent to which the as-is software can be used (i.e. ease of use, reliability and efficiency),
- maintainability, ease of identifying what needs to be changed. He defined testability, understandability and flexibility/modifiability as the sub characteristics of maintainability.
- portability, ease of changing software to accommodate a new environment.

Boehm’s Model



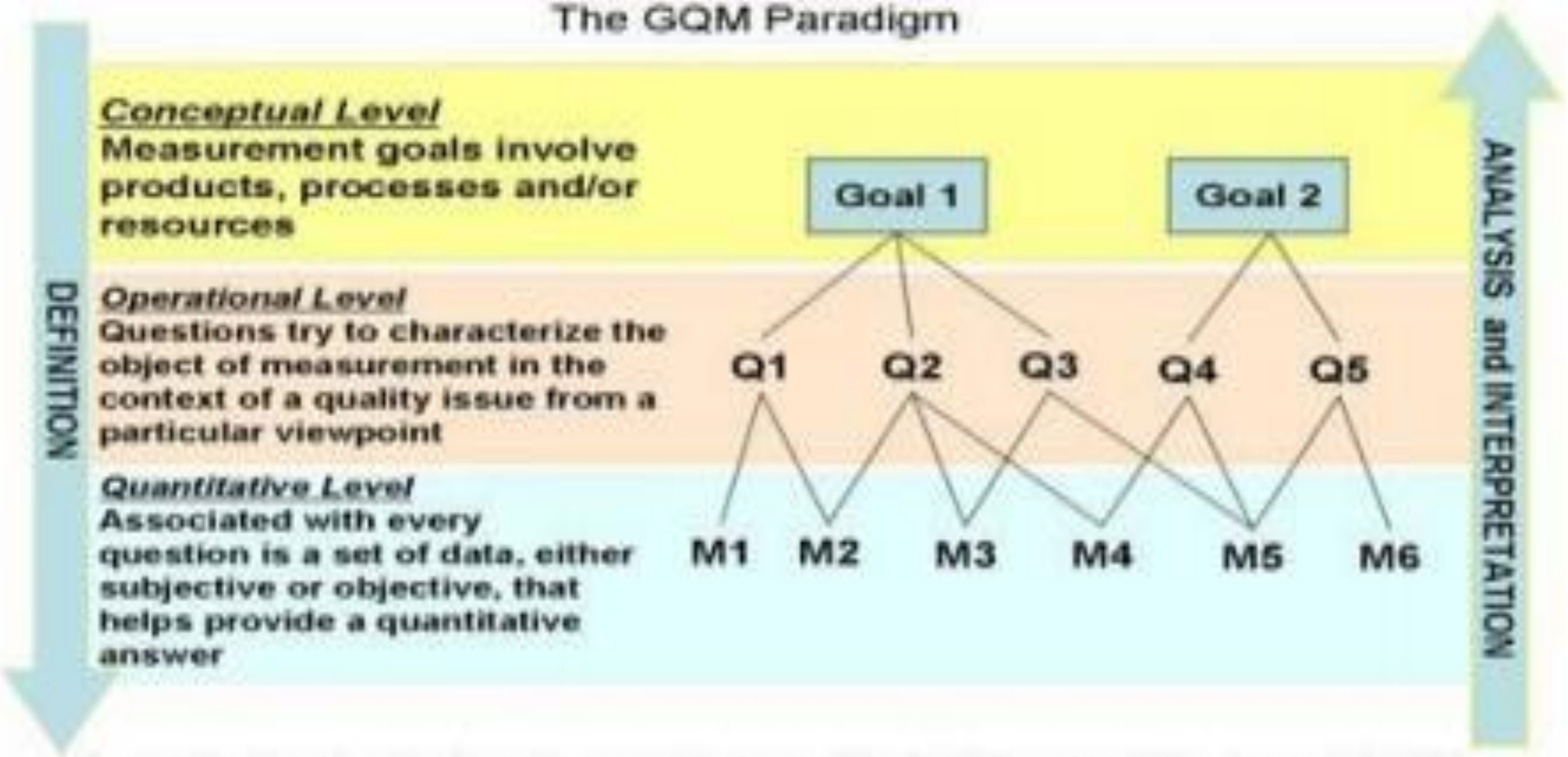
Boehm's Software Quality Characteristics Tree (Boehm et al. 1978)

GQM- Goal Question Metric

- GQM, the acronym for “Goal, Question, Metric", is an approach to software metrics
- GQM approach provides a framework with 3 steps:
  - Conceptual level(Goal):: List the major goals of the project/Process
  - Operational level(Question):: Derive from each goal the questions that must be answered to determine if the goals are being met
  - Quantitative level(Metric):: Decide what must be measured to answer the questions adequately

GQM- Goal Question Metric

The GQM Paradigm



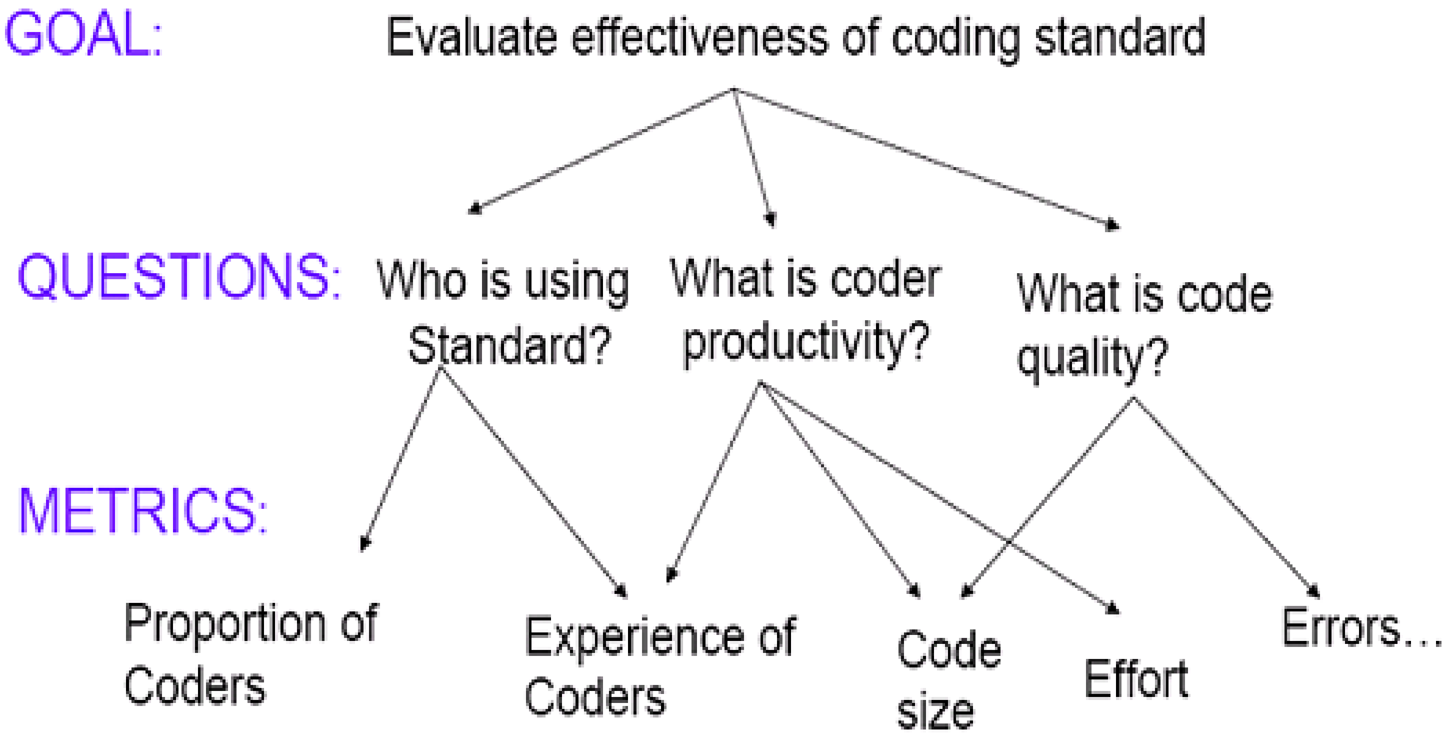
Conceptual Level
Measurement goals involve products, processes and/or resources

Operational Level
Questions try to characterize the object of measurement in the context of a quality issue from a particular viewpoint

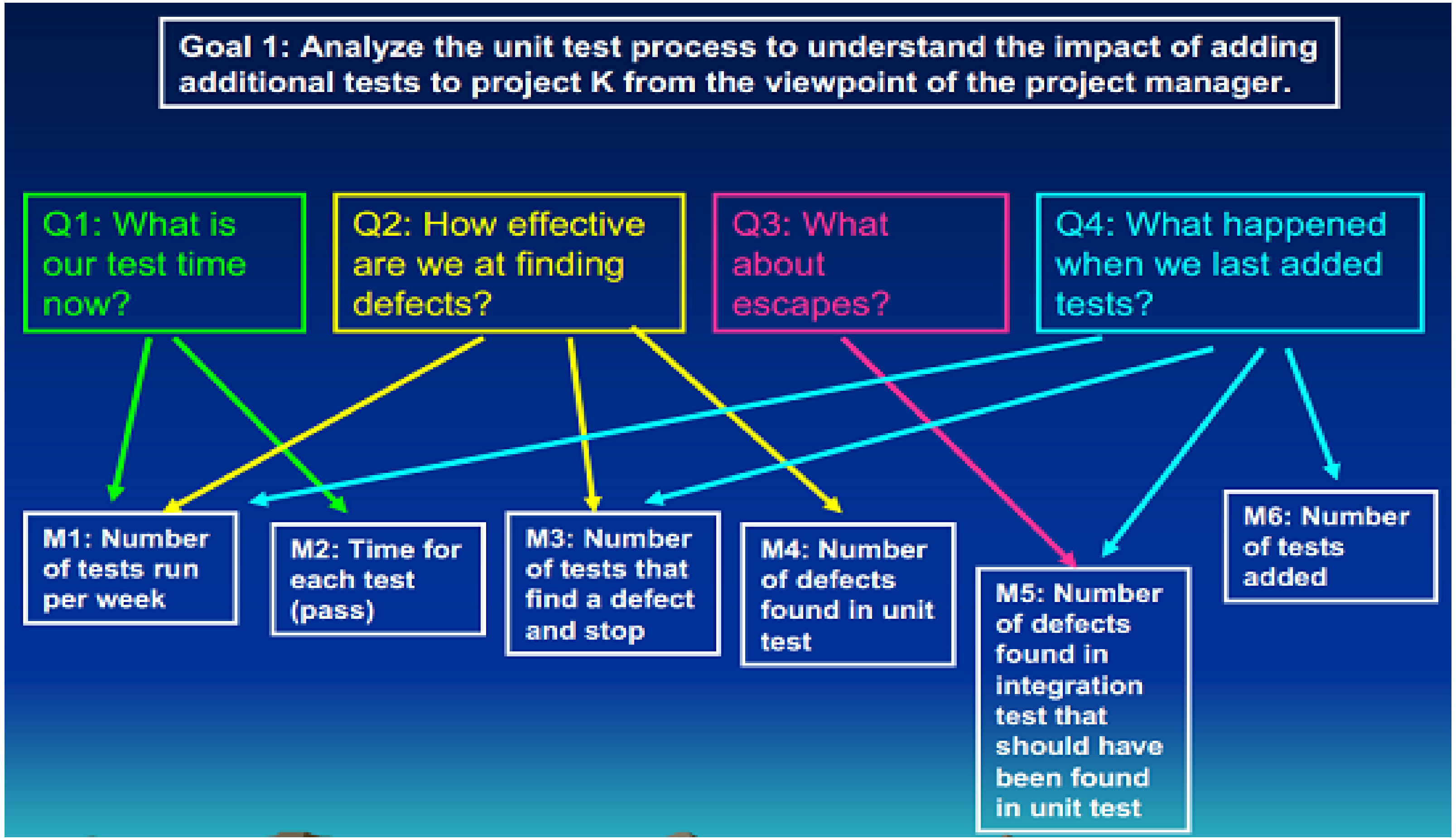
Quantitative Level
Associated with every question is a set of data, either subjective or objective, that helps provide a quantitative answer

Source: Derived from Basili, Caldiera, and Rombach, "The Goal Question Metric Approach", 1990

GQM- Example


GOAL:
Evaluate effectiveness of coding standard
QUESTIONS:
Who is using Standard?
What is coder productivity?
What is code quality?
METRICS:
Proportion of Coders
Experience of Coders
Code size
Effort
Errors…

GQM- Example



Goal 1: Analyze the unit test process to understand the impact of adding additional tests to project K from the viewpoint of the project manager.

Q1: What is our test time now?

Q2: How effective are we at finding defects?

Q3: What about escapes?

Q4: What happened when we last added tests?

M1: Number of tests run per week

M2: Time for each test (pass)

M3: Number of tests that find a defect and stop

M4: Number of defects found in unit test

M5: Number of defects found in integration test that should have been found in unit test

M6: Number of tests added

Cost of Quality

- Cost of quality includes all costs incurred in the pursuit of quality or to perform quality related work
- The "cost of quality" isn't just the price of creating a quality product or service. It's also the cost of NOT creating a quality product or service.
- Every time work is redone, the cost of quality increases. Obvious examples include:
  - ➢ The reworking of a manufactured item.
  - ➢ The retesting of a component.
- In short, any cost that would not have been expended if quality were perfect contributes to the cost of quality.

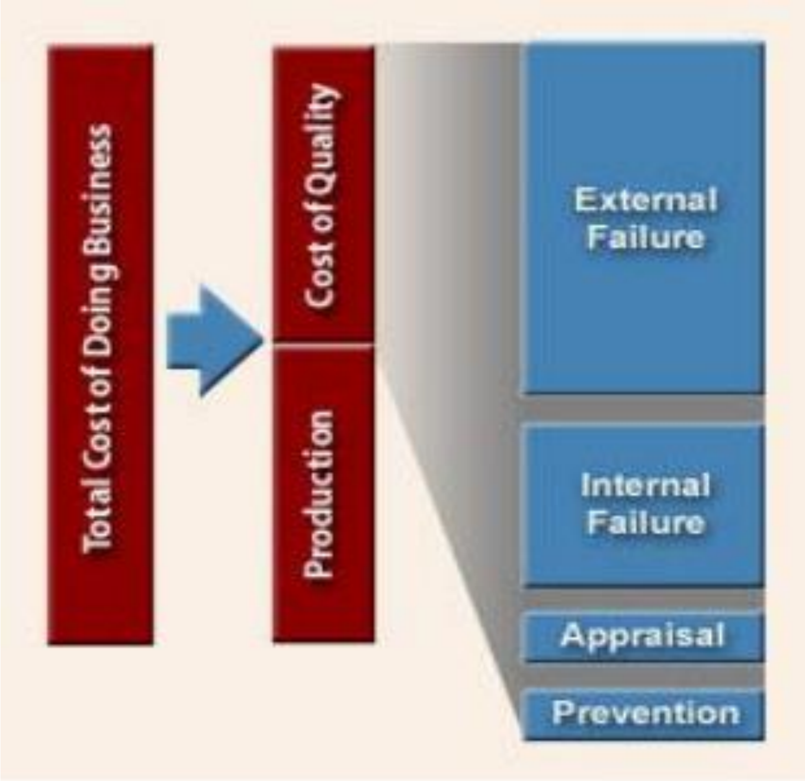
QUIZ-I (Time: 20 mins.)

1. Define Software quality, quality control and quality engineering.
2. Briefly describe the different views of quality.

Cost of Quality

Total Quality cost includes:

- **Prevention cost:**
- **Appraisal cost:**
- **Failure cost:**
  - **Internal failure cost**
  - **External failure cost**



Prevention Cost

- Generally the most effective way to manage quality costs is to avoid having defects in the first place.
- It is much less costly to prevent a problem from ever happening than it is to find and correct the problem after it has occurred.
- Prevention costs support activities whose purpose is to reduce the number of defects from occurring.
- Examples are the costs of:
  - ➢ quality planning
  - ➢ testing equipment
  - ➢ Staff training
  - ➢ Quality improvement process

    e.g. 6 Sigma

Appraisal Cost

- The costs associated with measuring, evaluating or auditing products or services to assure conformance to quality standards and performance requirements.
- Any defective parts and products should be caught as early as possible in the production process.
- Appraisal costs, which are sometimes called inspection costs, are incurred to identify defective products before the products are shipped to customers.
- These include the costs of:
  - ➢ Inspections
  - ➢ Testing
  - ➢ Maintenance of test equipment
  - ➢ Supervision of testing and inspection activities

Failure Cost

- The costs resulting from products or services not conforming to requirements/design specifications or customer/user needs. Failure costs are divided into two categories.
  - Internal Failure Costs
  - External Failure Costs

Failure Cost

- Internal Failure Costs
  - Internal failure costs result from identification of defects before they are shipped to customers.
  - The more effective a company's appraisal activities the greater the chance of catching defects internally and the greater the level of internal failure costs. This is the price that is paid to avoid incurring external failure costs, which can be devastating.
  - These include the costs of:
    - Scrap
    - Rework
    - Repair
    - Re-testing
    - Rejected products

Failure Cost

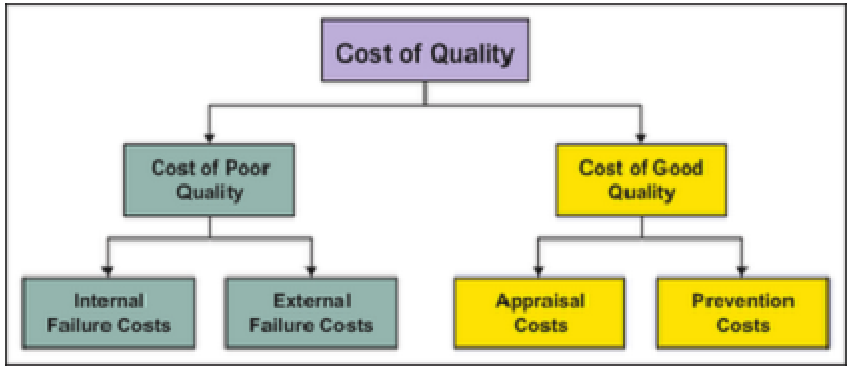
- **External Failure Costs**
  - ➢ Failure costs occurring after delivery or shipment of the product and during or after furnishing of a service to the customer.
  - ➢ These include the costs of:
    - ❑ Warranty
    - ❑ product return and replacement
    - ❑ help line support
    - ❑ Liability arising from legal actions against a company
- External failure costs usually give rise to another intangible cost. These intangible costs are hidden costs that involve the company's image. They can be 3 or 4 times greater than tangible costs. Missing a deadline, Lost sales arising from a reputation for poor quality or other quality problems can be intangible costs of quality.

Tangible vs Intangible Cost

- **What Does *Tangible Cost* Mean?**
  - **A quantifiable cost related to an identifiable source or asset. Tangible costs represent expenses arising from such things as purchasing materials, paying employees or renting equipment. They are prevention costs, appraisal costs, internal failure costs, and external failure costs.**
  - **Tangible cost is a cost that is seen instantly such as in purchasing products etc.**
- **What Does *Intangible Cost* Mean?**
  - An unquantifiable cost relating to an identifiable source. Intangible costs represent a variety of expenses such as losses in productivity, customer goodwill or drops in employee morale.
  - Intangible cost is a cost that is not seen but its effects are perceived later in future.
  - Intangible cost of an action may be much greater than tangible cost.

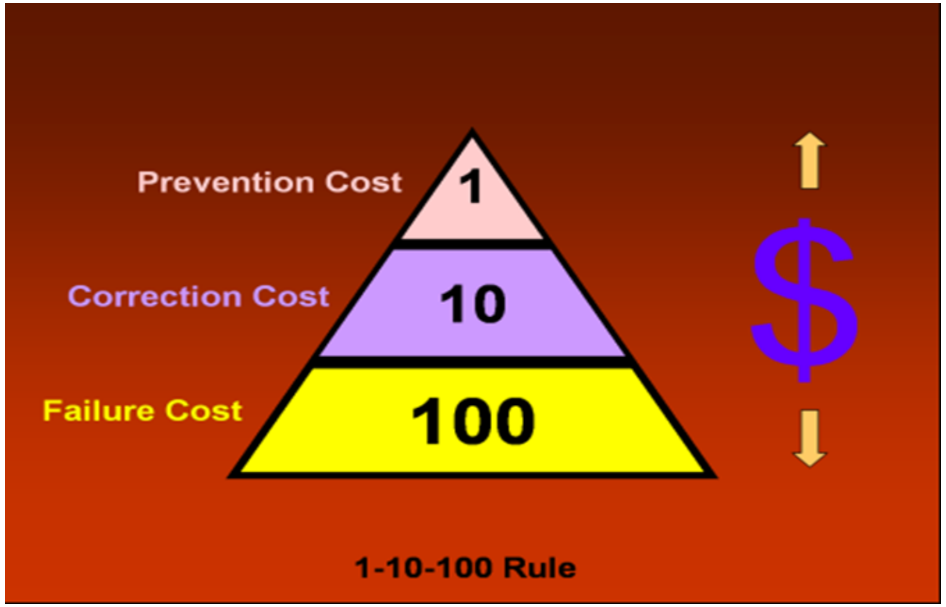
Failure Cost

- **Cost of Good Quality**
  - ➢The first two categories of cost are associated with putting systems and processes in place to reduce the likelihood of a failure.
- **Cost of Poor Quality**
  - ➢Internal failure costs, external failure costs and intangible cost that impair the goodwill of the company occur due to a poor quality so these costs are also known as **costs of poor quality** by some persons.



1 -10-100 Rule for Cost of Quality

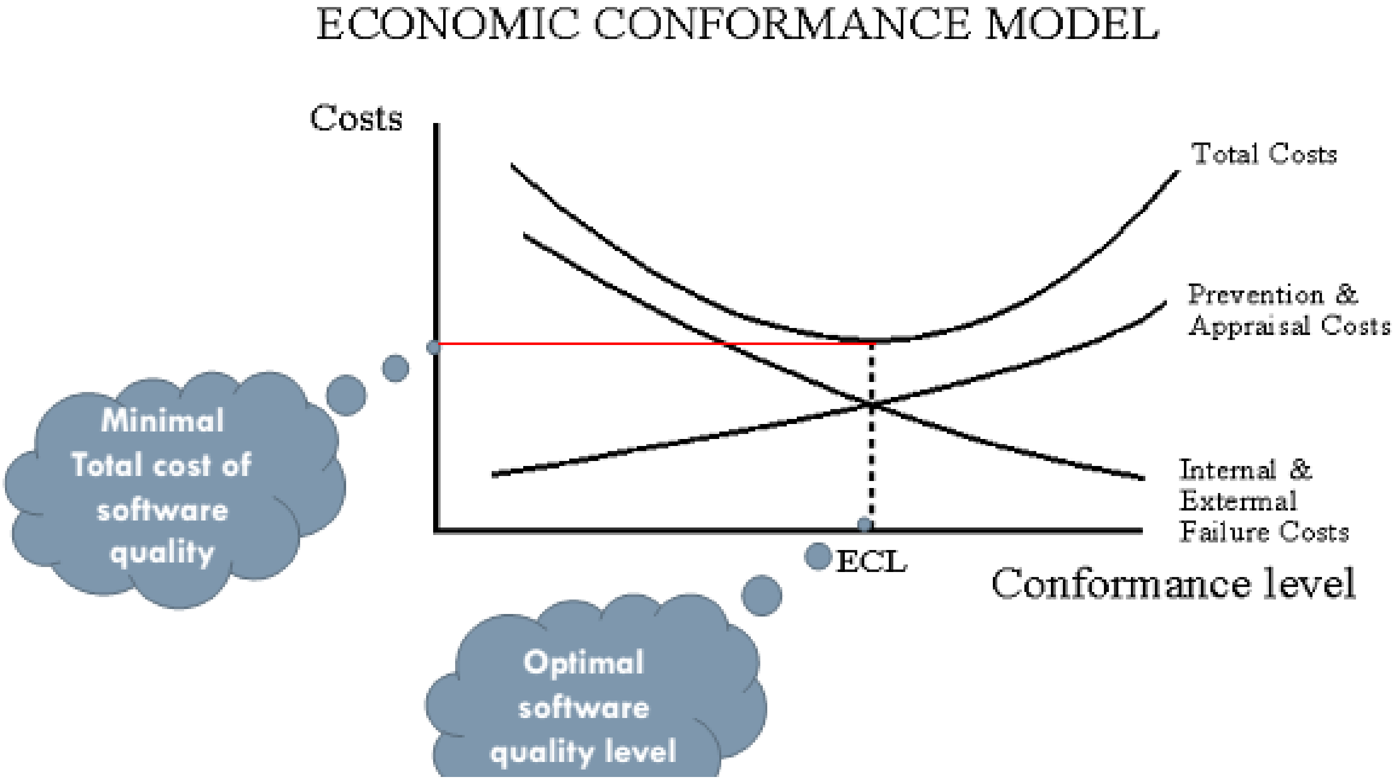
- The rule explains how failure to take notice of one cost escalates the loss in terms of dollars.
- 1 -10-100 rule shows that one dollar spent on prevention will save 10 dollars on correction and 100 dollar on failure costs.



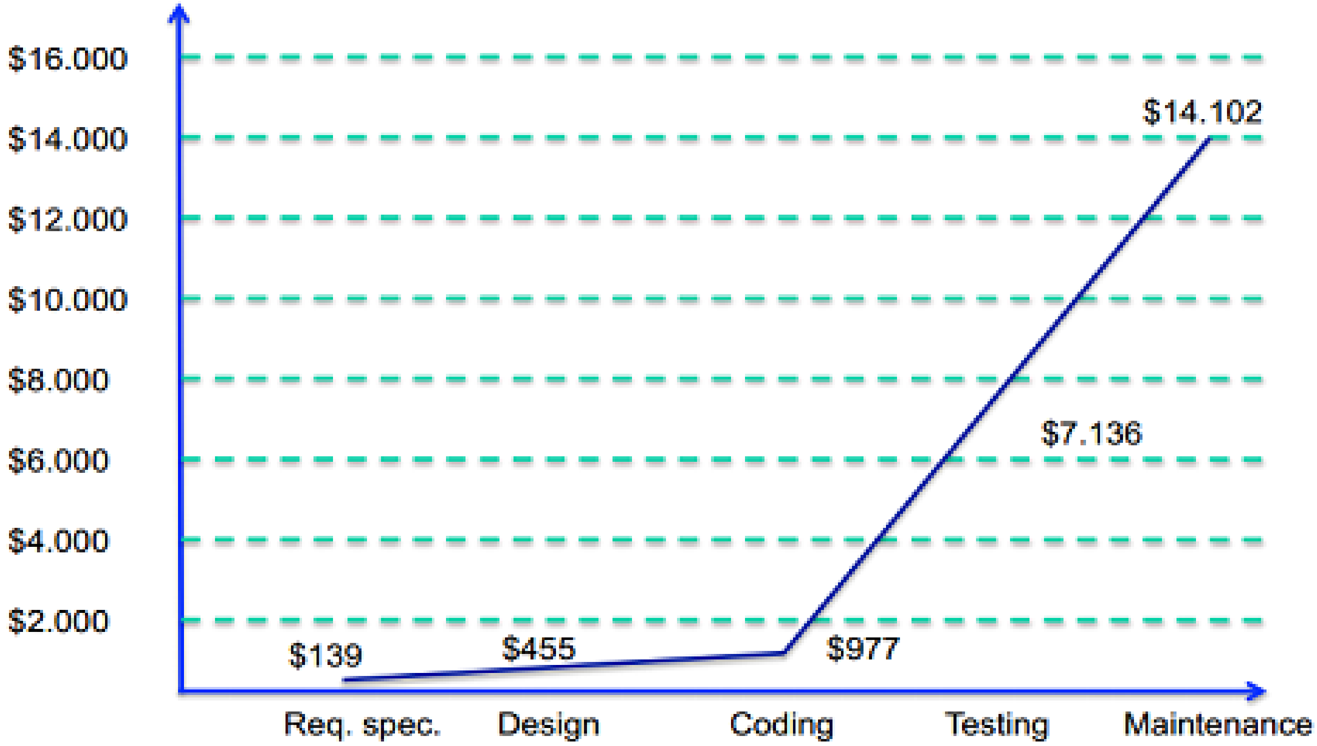
Quality Cost Conformance Model

- The quality cost conformance model provides an example of a constrained optimization approach. In this model the economic conformance level (ECL) is obtained where prevention and appraisal costs are equal to external and internal failure costs.
- Prevention and appraisal costs increase as the level of conformance of quality increases.
- Failure costs are expected to decrease as the level of conformance of quality increases. Therefore, the total costs associated with conformance of quality will be U-shaped as indicated in the figure.

The Quality Cost Conformance Model

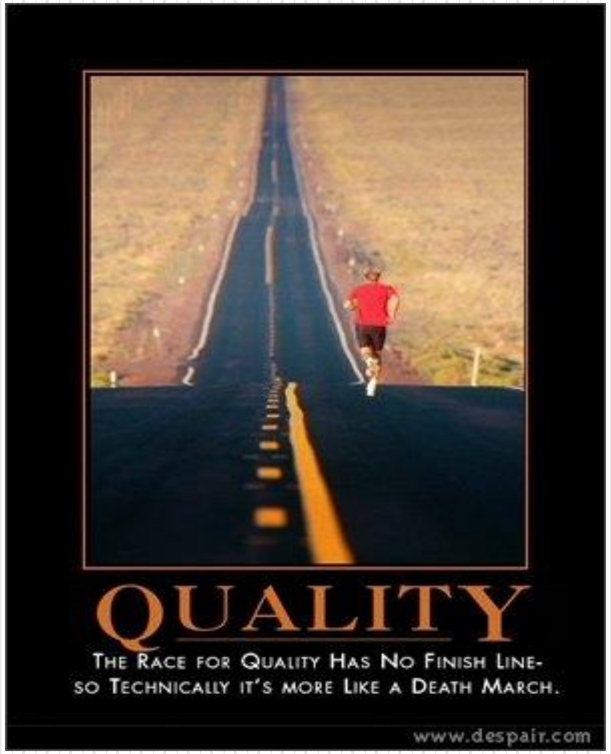


Repair Cost of Defects



Repair costs of defects (Pressman 2010, Boehm & Basili 2001)

The Race for Quality has no finish line, technically it’s more like a death march.


QUALITY
The Race for Quality Has No Finish Line-
so Technically it's more Like a Death March.
www.despair.com

References

1. http://www.cis.gsu.edu/~ghubona/cis8300/ISO9126.pdf
2. http://en.wikipedia.org/wiki/ISO/IEC_9126
3. http://www.bth.se/tek/besq.nsf/%28WebFiles%29/CF1C3230DB425EDCC125706900317C44/$FILE/chapter_1.pdf
4. http://wwwagse-old.informatik.uni-kl.de/pubs/repository/basili94b/encyclo.gqm.pdf
5. http://www.iteva.rug.nl/gqm/GQM%20Guide%20non%20printable.pdf
6. http://agileinaflash.blogspot.com/2009/04/furps.html
7. http://www.accountingformanagement.com/quality_costs.htm
8. http://totalqualitymanagement.wordpress.com/2009/02/25/what-is-1 -10-100-rule/
9. http://maaw.info/QualityCostConformanceModel.htm
10. http://www.architecting.co.uk/presentations/NFRs.pdf
11. http://www.authorstream.com/Presentation/cherry.jaspreet-982677-software-quality-modelsand-standars/
12. http://www.tol.oulu.fi/users/ilkka.tervonen/SQTe_2_11.pdf
13. http://www.differencebetween.com/difference-between-tangible-and-vs-intangible-cost/
14. http://www.investopedia.com/terms/t/tangible-cost.asp#axzz26q7MFRjD