Software Quality & Metrics

What Is Software Quality?

- Quality must be defined and measured if improvement is to be achieved.
- Quality is not a single idea, but multidimensional concept.
- Dimensions of quality include:
  - entity of interest,
  - viewpoint on that entity, &
  - quality attributes of that entity.

Contd…

- A popular view - it is an intangible trait
  - It can be discussed, felt, and judged, but cannot be weighed or measured.
- Crosby (1979) defines quality as "conformance to requirements" and
- Juran and Gryna (1970) define it as "fitness for use."

Contd…

- "Conformance to requirements" implies that requirements must be clearly stated such that they can't be misunderstood.
- In the development and production process, measurements are taken regularly to determine conformance to those requirements.
- The non conformances are regarded as defects—the absence of quality.

Contd…

- The "fitness for use" definition takes customers' requirements and expectations into account, which involve whether the products or services fit their uses.
- Since different customers may use the products in different ways, it means that products must possess multiple elements of fitness for use.
- The two most important parameters are:
  - *quality of design &*
  - *quality of conformance.*

Contd…

- Quality of design in popular terminology is known as grades or models.
- Quality of design can be regarded as the determination of requirements and specifications.
- Quality of conformance is conformance to requirements i.e. the extent to which the product conforms to the intent of the design.

Contd…

- From a customer's viewpoint, quality is the customer's perceived value of the product, based on a variety of variables such as price, performance, reliability, and satisfaction.

The Role of the Customer

- To achieve the state of conformance to USER requirements:
  - Customers' requirements gathered and analyzed
  - Specifications to meet those requirements must be produced, &
  - The product developed and manufactured accordingly.
- In each phase of the process, errors can occur that will affect the quality of the finished product.
- The requirements may be erroneous.
- The development and manufacturing process may be subject to variables that induce defects, and so forth.

Developer & Customer

- Customer's perspective - satisfaction after the purchase of the product is the ultimate validation that the product conforms to requirements and is fit to use.
- Producer's perspective, once requirements are specified, developing and producing the product in accordance with the specifications is the path to achieving quality.

Quality Levels

Two levels q & Q

- Intrinsic product quality, often operationally limited to the product's defect rate and reliability is "small *q*" *(q for quality).*
- *The broader definition of quality includes product quality, process quality,* and customer satisfaction, and it is referred to as the "big *Q*."

IBM - CUPRIMDSO

- IBM monitors satisfaction with its software products in levels of CUPRIMDSO:
  - Capability [functionality],
  - Usability,
  - Performance,
  - Reliability,
  - Installability,
  - Maintainability,
  - Documentation/information,
  - Service, and
  - Overall

HP - FURPS

- Hewlett-Packard focuses on FURPS:
  - Functionality,
  - Usability,
  - Reliability,
  - Performance &
  - Serviceability

Contd…

- To increase overall customer satisfaction:
  - The quality attributes must be taken into account in the planning and design of the software.
  - For example, the higher the functional complexity of the software, the harder it becomes to achieve maintainability.
- 15% or more of all software defects are requirements errors.

Quality Improvement

- We need models of the development process, and within the process we need to select and deploy:
  - specific methods & approaches &
  - employ proper tools and technologies.
- We need measures of the characteristics and quality parameters of the development process and its stages.
- Metrics and models to ensure that the development process is under control and moving toward the product's quality objectives.

Total Quality Management

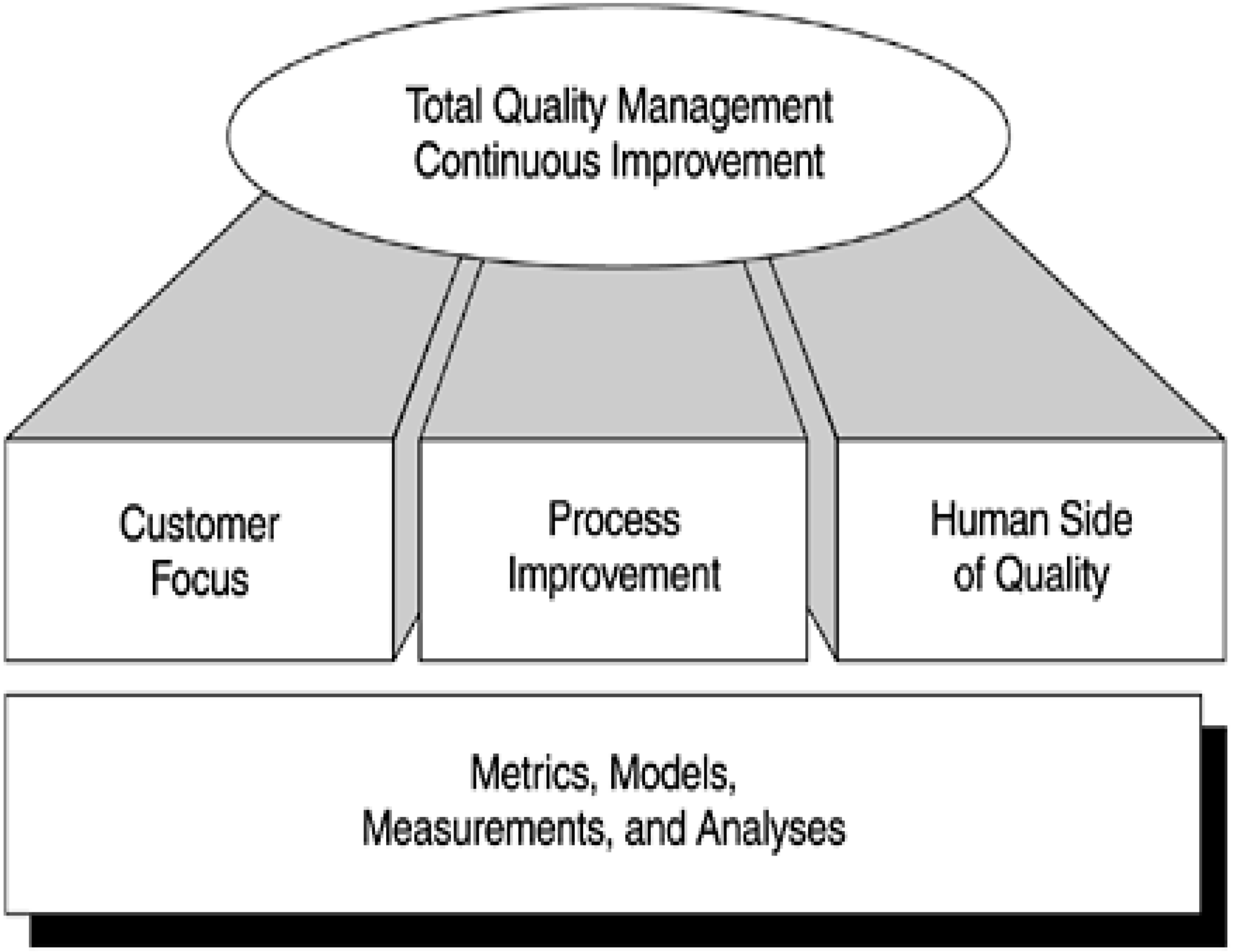
- It represents a style of management aimed at achieving long-term success by linking quality and customer satisfaction.
- In the computer and electronic industry, examples of successful TQM implementation include:
  - Hewlett-Packard's Total Quality Control (TQC),
  - Motorola's Six Sigma Strategy &
  - IBM's Market Driven Quality.

Key elements of a TQM

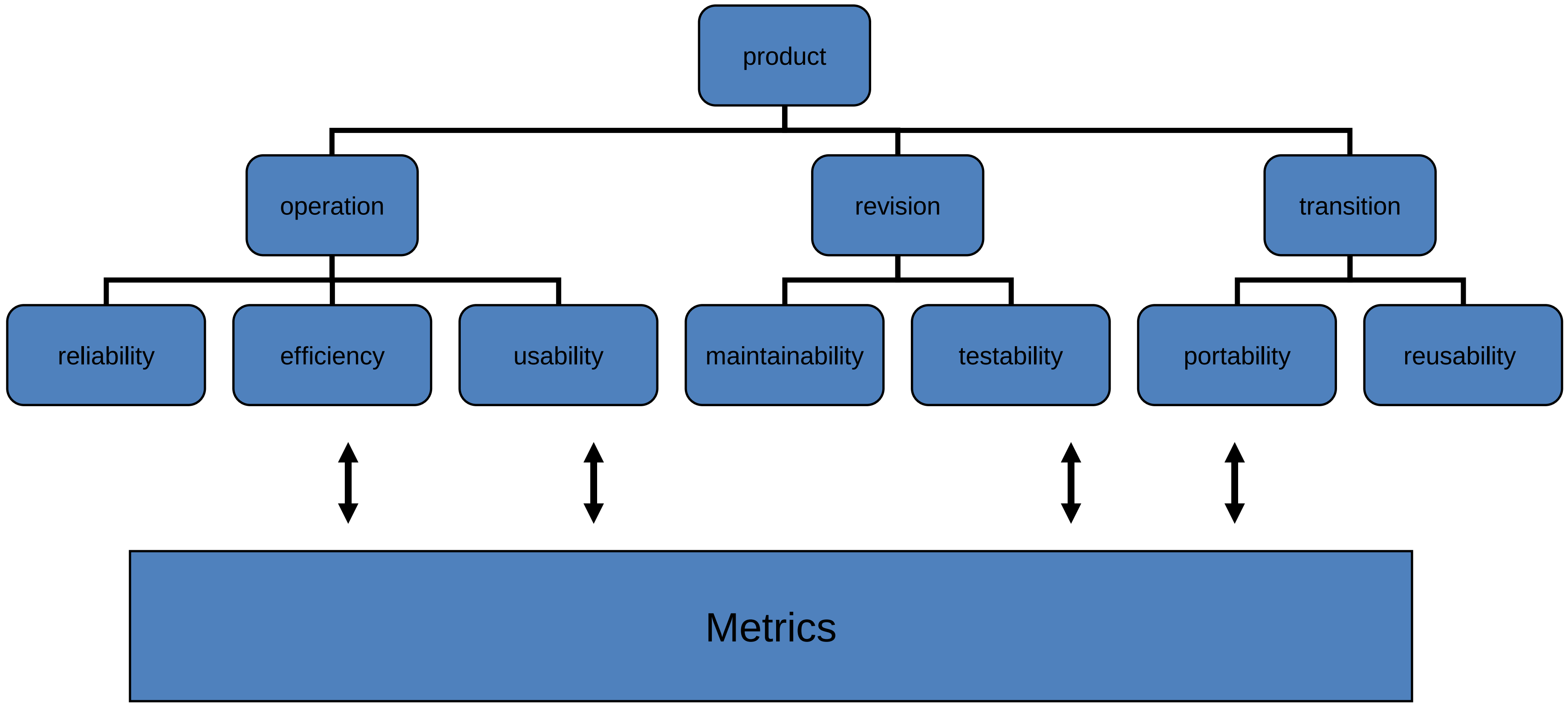
- *Customer focus*
    - *The objective is to achieve total customer satisfaction. Customer focus includes studying* customers' wants and needs, gathering customers' requirements, and measuring and managing customers‘ satisfaction.
- *Process*
    - *The objective is to reduce process variations and to achieve continuous process improvement.* This element includes both the business process and the product development process.

Contd…

- *Human side of quality*
  - *The objective is to create a companywide quality culture. Focus areas include* leadership, management commitment, total participation, employee empowerment, and other social, psychological, and human factors.
- *Measurement and analysis:*
  - *The objective is to drive continuous improvement in all quality parameters by* the goal-oriented measurement system.


Total Quality Management
Continuous Improvement
Customer Focus
Process Improvement
Human Side of Quality
Metrics, Models, Measurements, and Analyses

Quality Model



Quality Metrics

- Measures conformance to explicit requirements, following specified standards, satisfying of implicit requirements
- Software quality can be difficult to measure and is often highly subjective

1. Correctness:
   - The degree to which a program operates according to specification
   - Metric = Defects per FP
2. Maintainability:
   - The degree to which a program is amenable to change
   - Metric = Mean Time to Change. Average time taken to analyze, design, implement and distribute a change

Quality Metrics: Further Measures

3. Integrity:

- The degree to which a program is impervious to outside attack
- $$\sum_i (1 - t_i) \cdot s_i$$
- Summed over all types of security attacks, *i*, where *t* = threat (probability that an attack of type *i* will occur within a given time) and
- *s* = security (probability that an attack of type *i* will be repelled)

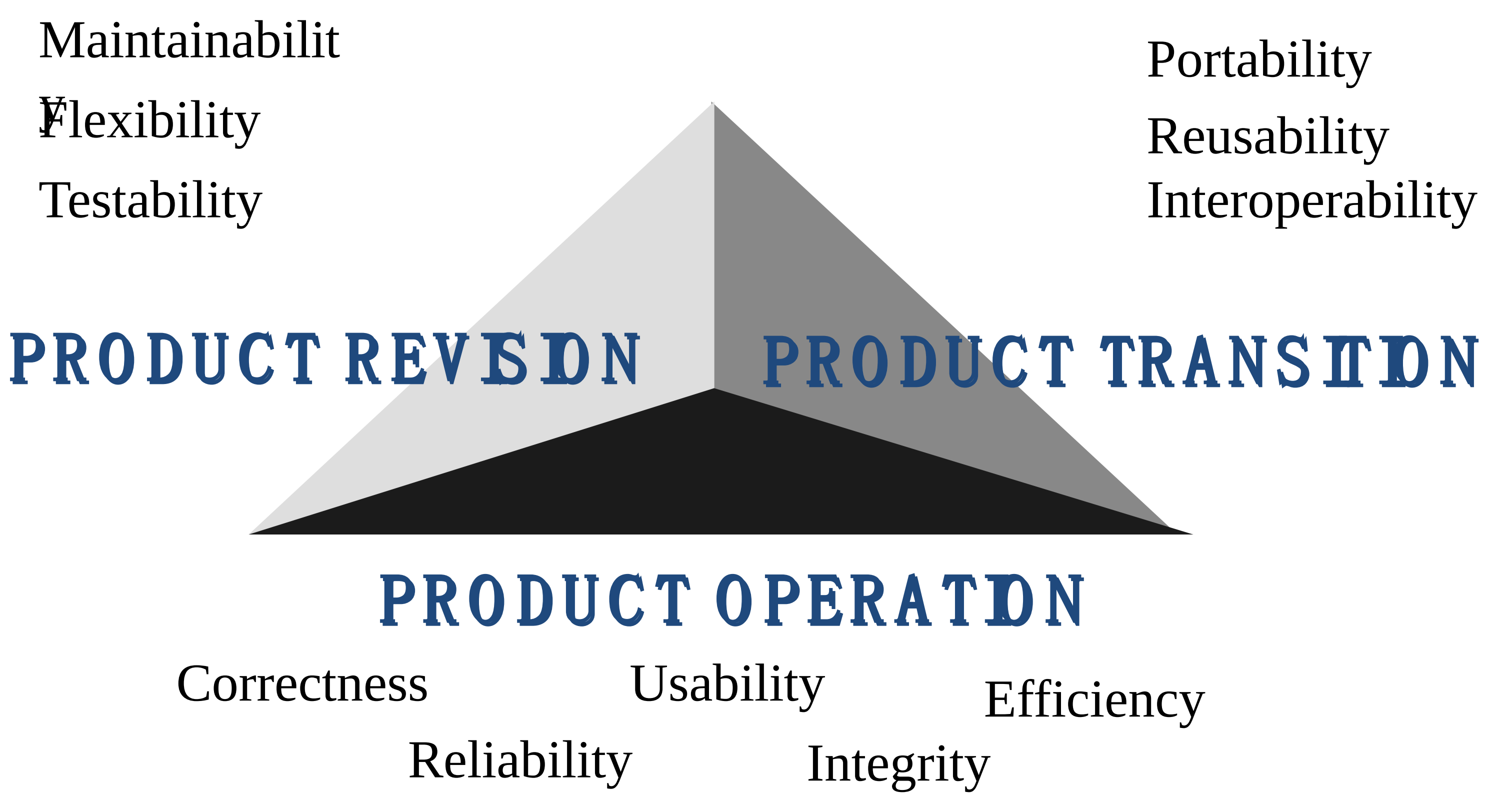
Contd…

4. Usability: The degree to which a program is easy to use.

   Metric Measures:

   – the skill required to learn the system,
   – the time required to become moderately proficient,
   – the net increase in productivity and
   – assessment of the users attitude to the system

Quality Metrics: McCall’s Approach



McCall’s Triangle of Quality

Quality Metrics: Deriving McCall's Quality Metrics

- Assess a set of quality factors on a scale of 0 (low) to 10 (high)
- Each of McCall's Quality Metrics is a weighted sum of different quality factors
- Example:
  - Correctness = Completeness + Consistency + Traceability
  - Completeness is the degree to which full implementation of required function has been achieved
  - Consistency is the use of uniform design and documentation techniques
  - Traceability is the ability to trace program components back to analysis
- This technique depends on good objective evaluators because quality factor scores can be subjective

ISO 9126 Software Quality Factors

- Functionality
  - The degree to which the software satisfies stated needs
- Reliability
  - The amount of time that the software is available for use
- Usability
  - The degree to which the software is easy to use

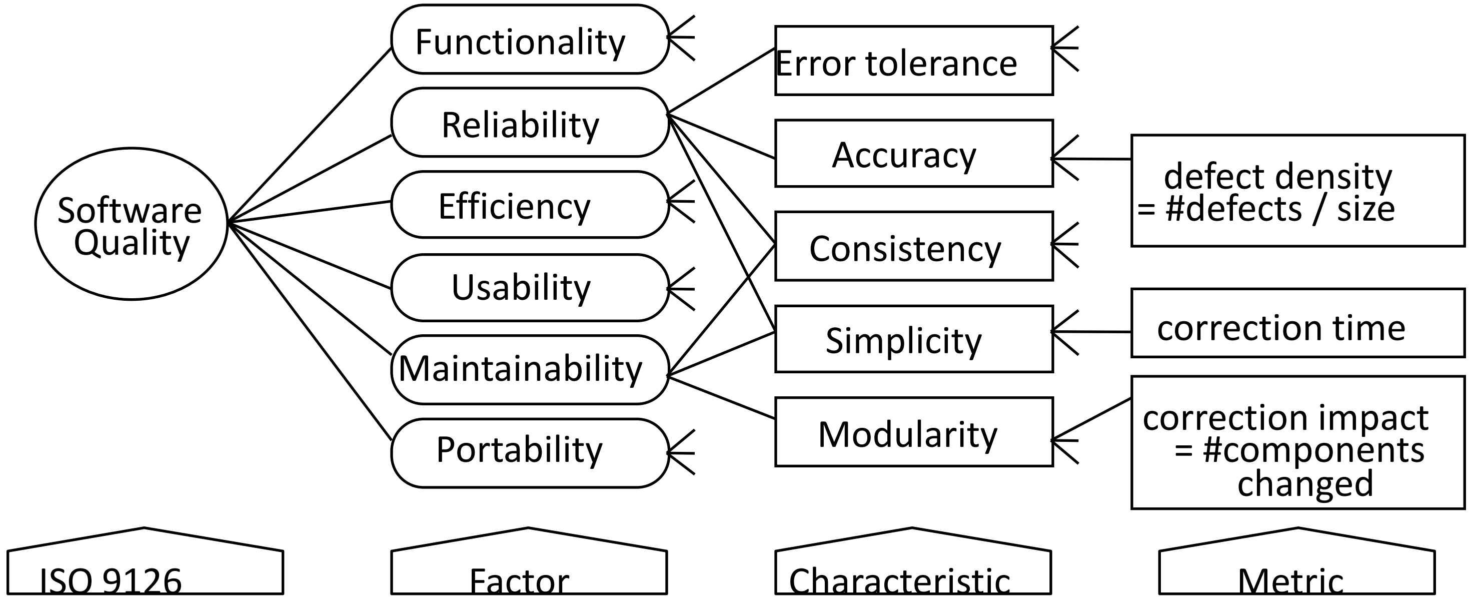
ISO 9126 Software Quality Factors

- Efficiency
  - The degree to which the software makes optimal use of system resources
- Maintainability
  - The ease with which repair and enhancement may be made to the software
- Portability
  - The ease with which the software can be transposed from one environment to another

Quantitative Quality Model

Quality according to ISO 9126 standard

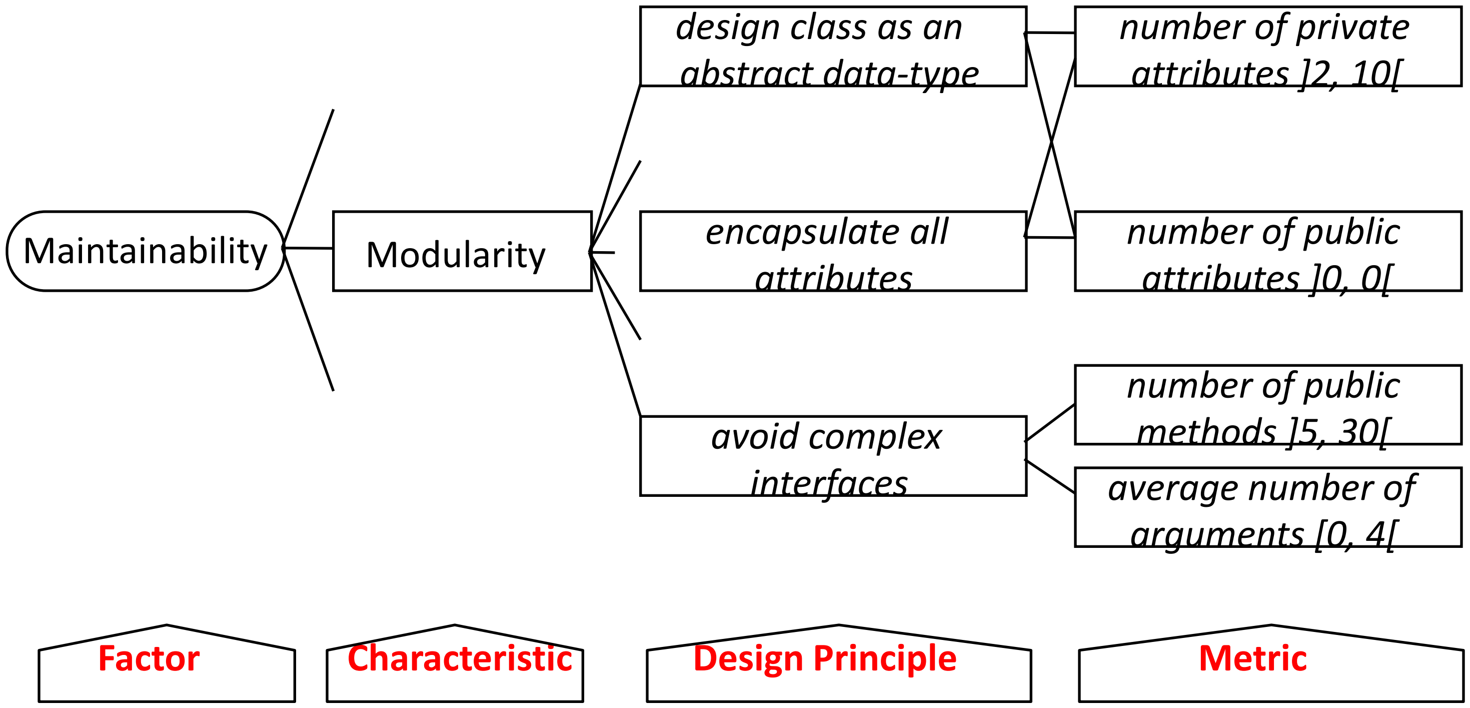
- Divide-and conquer approach via "hierarchical quality model"
- Leaves are simple metrics, measuring basic attributes



“Define your own” Quality Model

Define the quality model with the development team

- Team chooses the characteristics, design principles, metrics ... and the thresholds


Maintainability
Modularity
design class as an abstract data-type
encapsulate all attributes
avoid complex interfaces
number of private attributes]2, 10[
number of public attributes]0, 0[
number of public methods]5, 30[
average number of arguments [0, 4[
Factor
Characteristic
Design Principle
Metric

Sample Quality Metrics (I)

Productivity (Process Metric)

- functionality / time
- functionality in LOC or FP; time in hours, weeks, months
  - be careful to compare: the same unit does not always represent the same
- Does not take into account the quality of the functionality!

MTTF

Mean Time To Failure (MTTF)

- It is the average time that elapses until a failure occurs.
- It does not provide information about the distribution of the TTF,
- Hence we need to estimate the variance of the TTF.

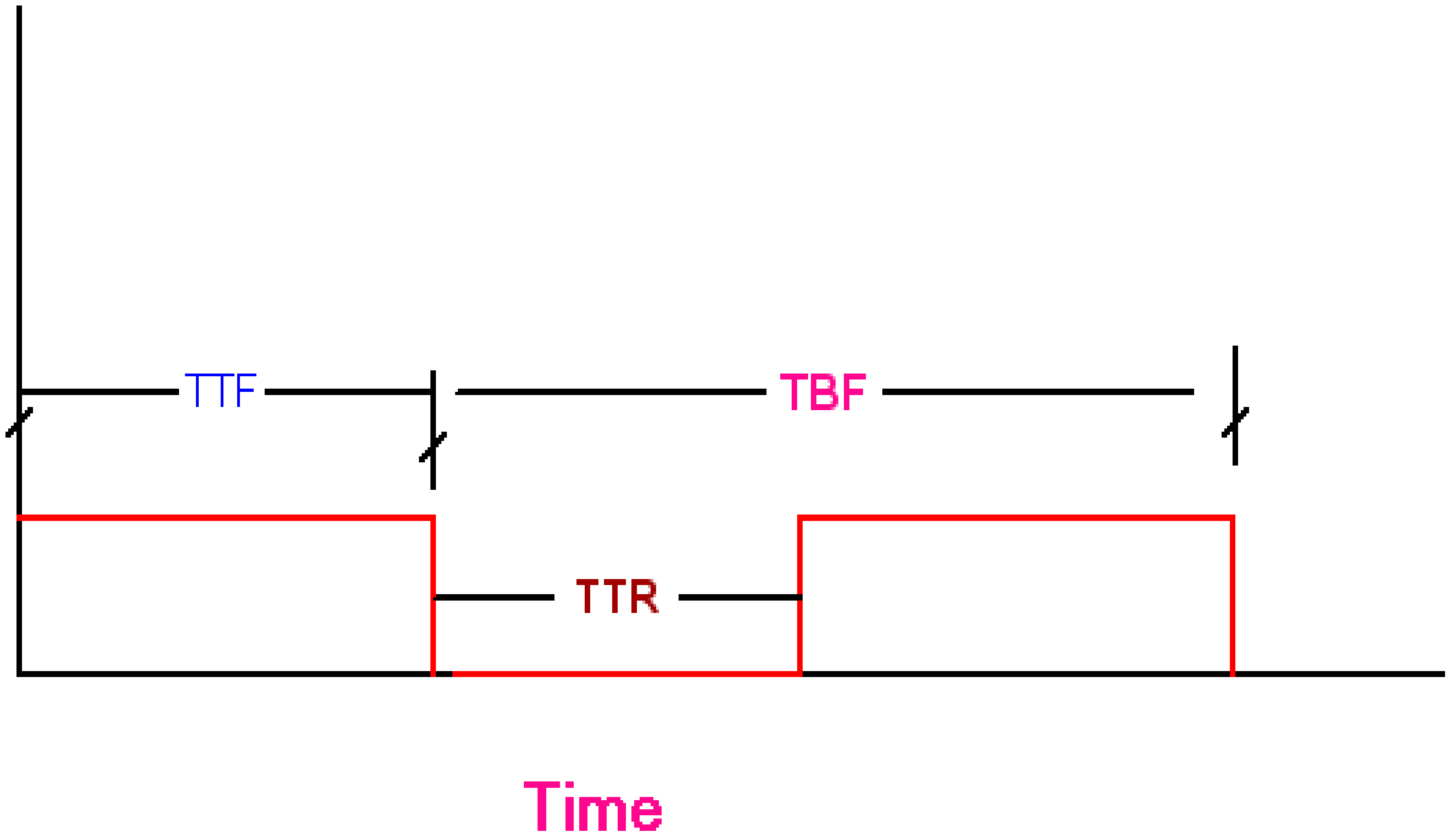
30

MTBF

Mean Time Between Failure (MTBF)

- It is the average time between successive failures. It is used for repairable systems.
- It provides the basis for estimating the number of failures in a given period of time.

Mean Time Between Failure: MTBF



32

Reliability (Product Metric)

- **MTTF (mean time to failure)** = average time between failures = # failures / time
  - time in execution time or calendar time

- **MTBF (mean time between failure)** = MTTF + mean time to repair
  - to know when your system will be available, take into account repair

Sample Quality Metrics (III)

Correctness (Product Metric)

- "a system is correct or not, so one cannot measure correctness"
- defect density = # known defects / product size
  - product size in LOC or FP
  - # known defects is a time based count!
- *do not compare across projects* unless your data collection is sound!

Metrics for Software Quality

The Primary source of quality measurers at the Project-level is ‘Errors and Defects’.

Metrics derived from Errors and Defects provide an indication of the effectiveness of Software Quality assurance and Control activities.

Error data can also be used compute the ‘Defect Removal Efficiency (DRE)’

QUALITY METRICS that are derived from Errors / Defects

- Product Errors Per Function Point
- Errors uncovered per review hours
- Errors uncovered per Testing hour

Measuring Quality

There are many Measures of Software Quality; But the following Four measures provide useful indicators for the Project Team.

- Software Correctness
- Software Maintainability
- Software Integrity
- Software Usability

Metrics for Software Quality

Correctness is the degree to which the software performs its required function.

<u>Most common measures for Correctness is:-</u>

- *DEFECTS / KLOC,*

Defects are these problems that are reported by Users after the Software has been released for general use.

For Quality assessment, Defects are counted over a standard period of time, typically for one year.

Software *Maintainability*

- Maintainability is the ease with which a program can be corrected when an error is encountered, adapted if its environmental changes, or enhanced if the customer desires a change in requirements.

There is no way to measure Maintainability directly so we must use indirect measurements.

- Mean Time To Change -(MTTC), which is a simple Time-oriented Metrics can be used to Analyze the changes required, Design the appropriate modification, Test, Implement and distribute the changes to all users.

- Programs that are maintainable have a lower MTTC than the programs that are not maintainable.