

# Software Quality (and ISO 25010)

Based on slides from Giuseppe Santucci

<http://www.sqa.net/>



How the customer explained it



How the Project Leader understood it



How the Analyst designed it



How the Programmer wrote it



How the Business Consultant described it



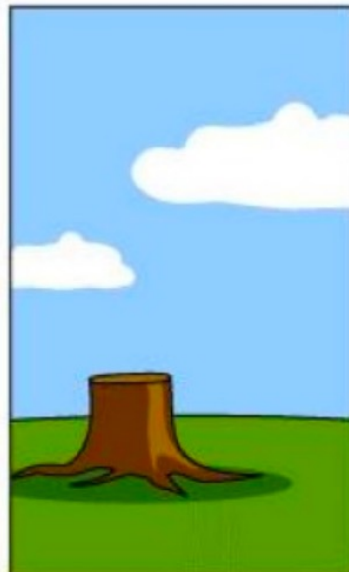
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

# Software ?

- ISO 12207 e IEEE definitions:
- Sw product: "set of programs, procedures, and documentation and associated data"
- Sw component: "identifiable part of a sw product, at intermediate stage or at the final stage of development"

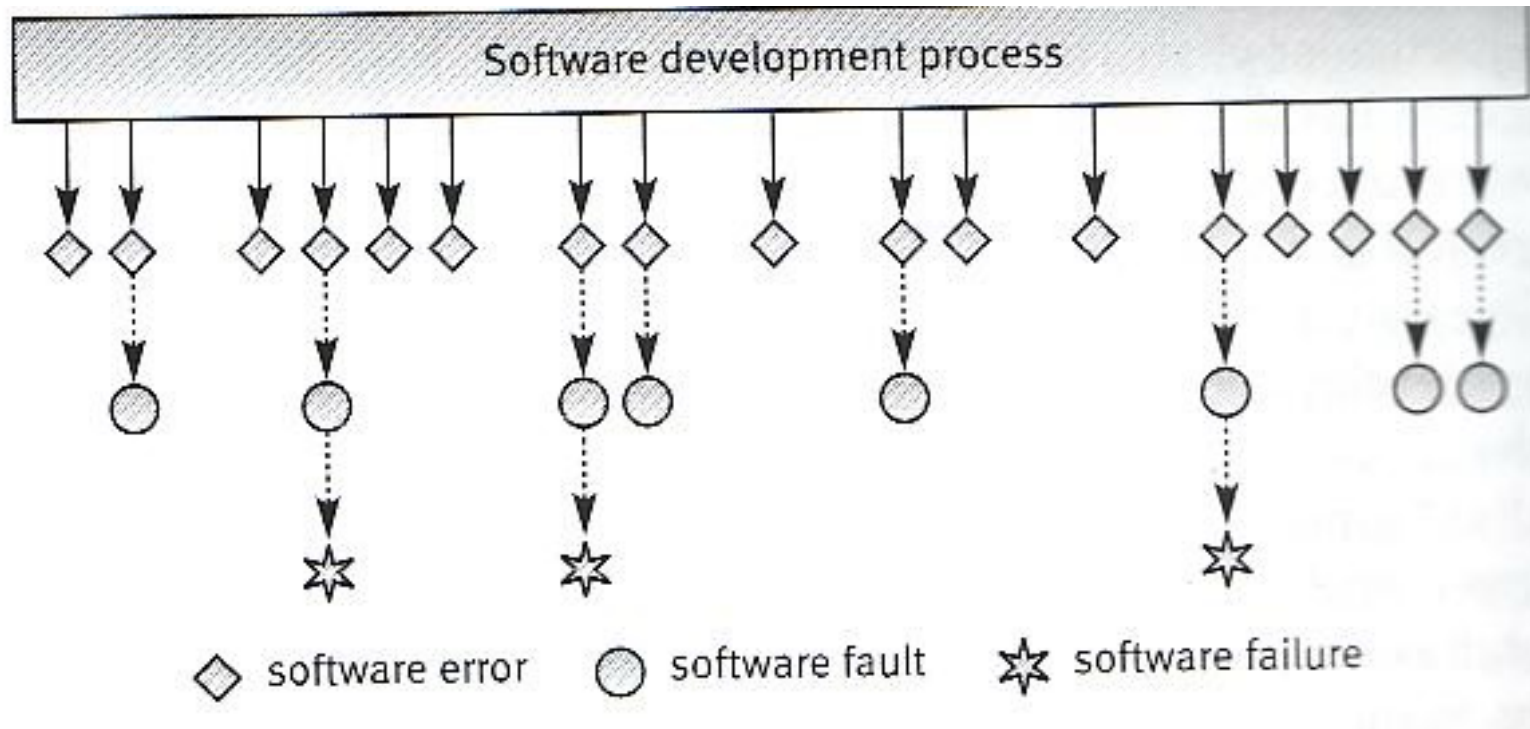
# 4 components

1. Code...
2. Procedures: all the needed information for the proper sw execution (sequence, people, etc.)
  - Activity order
  - Involved people, roles
3. Documentation
  - Developers (requirements, project, code description)
  - Users (user manual / on line help)
  - Maintainers (high-level documentation, useful to search error causes/points for changes)
4. Associated data (configuration files, data files (DBMS), test report)
  - Code quality is only one out 4 factors!

# Errors, fault (or defect), and failures

- The **error** is committed by a human being (in the code, documentation, data, procedures)
- The **fault** (defect) is a physical characteristic of a piece of code, a section of the documentation, a portion of the data, in the procedures, that can occur due to an error
- The **failure** is the result of a defect that can occur during the use of the software product and is perceived by the user

# Errors, fault (or defect), and failures



Which implies that SW failures may appear after years of correct operations (case: year 2000 problem (1999?))

## MAINFRAME COBOL FILE STATUS KEYS / CODES:

File status Codes beginning with a '0' are considered  
Codes beginning with a '3' are considered "Permanent"

Code	Statements	Description of the Code
------	------------	-------------------------

99

READ

WRITE

REWRITE

DELETE

Record Locked by another user- record access failed.

# Error sources (1,2)

1. Defective requirements
  - Wrong
  - Missing
  - Incomplete
  - Useless
2. Customer-developer misunderstanding of
  - Official requirements
  - Official changes
  - Verbal changes
  - Query-answer (originated by the developer)



# Error sources (3,4)

3. Deliberate deviation from requirements
  - SW reuse without correct analysis
  - Official omissions (time / money motivation)
  - Non official omissions (time / money motivation)
4. Design errors
  - Algorithms
  - Process misunderstanding (often associated with different customer-analyst cultures)
  - Errors on boundary conditions
  - Internal state omissions (no defined behavior for some situation)
  - Missing defined behavior for wrong input

# Error sources (5,6)

## 5. Coding errors

- ...

## 6. Non standard coding. Not a real error but it increases the number of errors number:

- Integrating non standard with standard modules
- SW reuse or changing programmers
- Revision
- Test
- Maintenance

# Error sources (7,8)

7. Reduced test phase (money, time, wrong planning)
  - Incomplete test plan
  - Missed or inadequate test error notification
  - Missed or incomplete error fix
8. Documentation errors
  - User Manual: functionality description omissions, non-existing functionality description, wrong or obsolete documentation
  - Design document errors / sw documentation errors
  - User documentation errors

# Source code errors

- During the inspection process a list of the most common mistakes is typically used
- The list should not include aspects of programming style (e.g., "there are enough comments?")
- Here we see a list ([Myers 1979]) that proposes an error classification that is programming language independent (even if some types of errors may not be significant in Java, but only in other languages, e.g., the C)
  - Data reference errors / Data-declaration errors
  - Computational errors
  - Comparison errors
  - Control-flow errors
  - Interface errors
  - I/O errors

# Source code errors (2)

## Data reference errors / Data-declaration errors:

- Are we referring to a not initialized variable ?
- The array index is within the correct values? “*off by one*” error? (Java and C...)
- The pointed memory has been correctly allocated? (*dangling reference problem*)
- Variable types are correct?
- There exist variable with similar names (very dangerous practice)?

# Source code errors (3)

## Data reference errors:

```
int i, sum;    (sums up the first 100 integers)
```

```
for (i=1; i<=100; ++i)
```

```
    sum = sum+i;
```

-----

```
int i, v[10];
```

```
for (i=1; i<=10; ++i)
```

```
    v[i] = 0;
```

-----

**Data reference errors /**  
**Data-declaration errors**

Computational errors

Comparison errors

Control-flow errors

Interface errors

I/O errors

# Source code errors (4)

## Computational errors:

- Do calculations involve inconsistent types (e.g., strings and integers)? There are inconsistencies in mixed calculations (e.g., integers and reals)
- Are there calculations involving compatible types but with different precision?
- In an assignment, the left value has less accurate representation of the right value ?
- Is it possible to get an overflow or underflow (e.g., in the intermediate calculations or conversions)?
- Division by zero?

Data reference errors /

Data-declaration errors

**Computational errors**

Comparison errors

Control-flow errors

Interface errors

I/O errors

# Source code errors (4)

## **Computational errors (continued):**

- Do expressions that contain various arithmetic operators, make right assumptions about the operator order and precedence?
- Does the value of a variable stay within a reasonable range? (e.g., the weight should be positive, ...)
- Are we adequately taking into account the finite precision of reals and integers ?



# Source code errors (5)

## Computational errors (continued):

```
-----  
int i,j,k;  
j=1;  
k=2;  
i=(j/k)*2;  
k=k/i;
```

```
-----  
long height=1000;
```

.... 10.000 Division/multiplication on height (simulation of the falling of an object in the air with friction, air density, speed, etc.);

if (height==0) ....

# Source code errors (6)



- **Computational errors:**
- On June 4, 1996 the Ariane 5 rocket launched by the European Space Agency exploded after about 39 seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing \$7 billions. The destroyed rocket and its cargo were valued at \$500 million. Why did it happen?
- The inertial reference system (gyroscope, accelerometers, etc..) was handled by redundant HW (two computers) and its goal was to provide the steering system with all the relevant values for the first 40 seconds of flight. Among these values, the horizontal velocity of the rocket with respect to the platform was converted by a 64 bit real to a 16 bit signed integer (**float to integer**);
- After about 36 seconds that number became greater than 32767, the largest integer storable in a 16 bit signed integer, and thus the conversion failed. The control was taken by the second computer that did the same calculations ending with the same error
- The second computer returned the overflow code error and performed a shutdown
- The steering system interpreted the **error code** as a bizarre flight condition and reacted quickly asking for an abrupt (and not needed) turning
- Because of the acceleration and the aerodynamic forces the engine started to detach from the rocket
- Sensors detected this dangerous situation and trigger the self-destruct mechanism...

The most expensive  
firework of the hystory



Is it possible to get an overflow or underflow (e.g., in the intermediate calculations or conversions)?

# Source code errors (7)

## Comparison errors:

- Are we comparing different variable types?
- Are we using the right comparison operators? (e.g., < vs <=)
- Are the Boolean expressions correct? (and, or, not usage)?
- Is the evaluation order of the expression correct? (in doubt, use parentheses...)
- Are Boolean operands Booleans?

Data reference errors /

Data-declaration errors

Computational errors

**Comparison errors**

Control-flow errors

Interface errors

I/O errors

# Source code errors (8)

## Comparison errors:

```
-----  
int i,j;  
j=1;  
scanf("%d",&i);  
if (i=j) printf("i e j are equals");  
-----
```

```
int grade;  
scanf("%d", &grade);  
/* from the keyboard you input 25 . Chose your preferred code below...*/  
if (17< grade< 32) printf ("exam is passed");  
    else printf ("exam is not passed");  
  
if (32 > grade >17) printf ("exam is passed");  
    else printf ("exam is not passed");
```

Data reference errors /  
Data-declaration errors  
Computational errors  
**Comparison errors**  
Control-flow errors  
Interface errors  
I/O errors

# Source code errors (9)

**Control-flow errors:** (often generated by a bad algorithm)

- Do cycles end?
- There exists unreachable code?
- Are the cycles termination conditions correct?
- “*off by one*” errors? (e.g., a cycle is executed once less or once more)
- Have been all the possible alternatives adequately considered? (e.g., a malformed switch instruction)

# Source code errors (10)

## Control flow errors:

```
long computeAverage(){
    int i,j;
    float average=0;
    i=0;
    FILE *f;
    f=fopen("dati.txt",r);
    do
        fscanf(f,"%d",&j);
        average=average+j;
        ++i;
    until !feof(f);

    return average/i;}
```

Data reference errors /  
Data-declaration errors  
Computational errors  
Comparison errors  
**Control-flow errors**  
Interface errors  
I/O errors

# Source code errors (11)

## **Interface errors:**

- Are the number and the order of parameters the same as the formal parameters?
- Are the assumptions on the type conversion between actual and formal parameters correct?
- Are parameters handled in the right way? (value or reference) ?
- Is a parameter involuntarily changed?
- Are the measure units used in a coherent way?(e.g., meters and yard)?

# The Gimli glider



- On 23 July 1983, Air Canada Flight 143, a Boeing 767-200 jet, ran out of fuel at 7920 m altitude
- Captain Pearson (an experienced glider pilot), helped by first officer Quintal that made vital real time calculations, piloted the Boeing at the optimal speed and eventually landed on the closed airport Gimli, with no consequence for the Boeing and passengers (just some broken legs)



Why did they ran out of fuel?

For a mechanical problem,  
the plane was refilled manually and the  
pilots had to convert manually the volume  
in weight

At the time of the incident,  
Canada was moving to  
the metric system

Manuals were reporting the math for  
manually converting **gallons in pounds**, not **liters in kg**  
The plane had less than half the required amount of fuel



Error in the user  
documentation...

Are the measure units used  
in a coherent way?



# Source code errors (12)


## I/O errors:

- Is the read from file done respecting the format?
- Are files opened with the right mode (read/write)?
- Are EOF conditions handled in the right way?

Data reference errors /  
Data-declaration errors  
Computational errors  
Comparison errors  
Control-flow errors  
Interface errors  
**I/O errors**

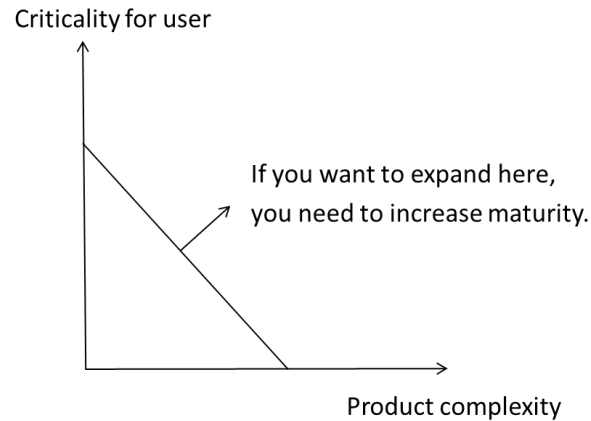
# Source code errors (13)

## Other:

- Are there declared but not used variables?
- Have been the warning messages adequately considered?
- Is the program robust enough? Does it control the input validity? 
- Have all the planned features for the program been implemented?

# QUALITY

# Quality work in a software company is done for different reasons



Take new challenges

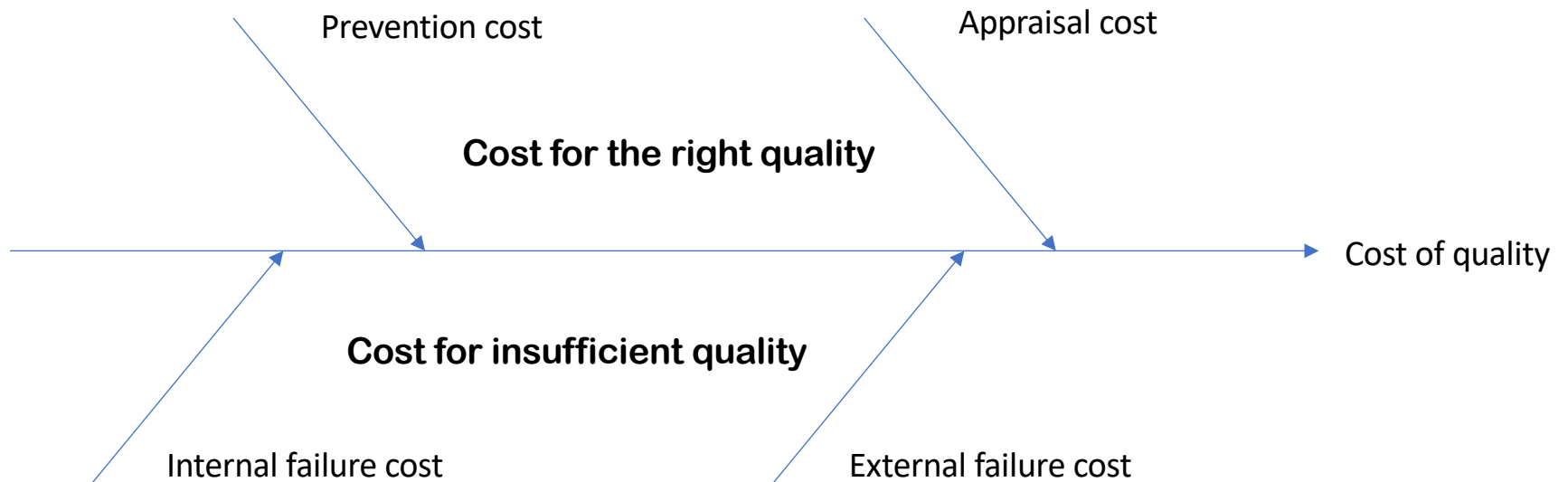


Grow your assets



Avoid complaints

# Quality is free?



[Feigenbaum, Armand V.](#) (November–December 1956), "Total Quality Control", [Harvard Business Review](#), **34** (6)

# Drivers for quality



# What does Quality mean to you?

- You are a project manager of the BlaBla communication AB in Sweden. Your company develops networking products such as routers and switches. Your company also develops software for routers and switches to make it easy to use for customers. You are competing with other 5 companies to get the contract for 1 million devices for third world countries as part of United Nation campaign. This contract is very important for your financial planning otherwise you can go bankrupt. The contract is to submit the following documentation of one line by just filling the blanks. Your answer (i.e. for the blank) must be between 1-5 words.

Your hardware and/or software has a good quality because it \_\_\_\_\_!

We offer three kinds of service:

**GOOD - CHEAP - FAST**

You can pick any two

**GOOD** service **CHEAP** won't be **FAST**

**GOOD** service **FAST** won't be **CHEAP**

**FAST** service **CHEAP** won't be **GOOD**

U NEAK SIGNS



# SW quality?

- IEEE:
  - The level at which a system, component or process meets the requirements
  - The level at which a system, component or process meets the needs and expectations of a user
- PRESSMAN:
  - Accordance with the functional and nonfunctional requirements, with explicit standards development, and features of a professionally developed software

# SW quality assurance (SQA):

- IEEE
  - a planned and structured actions necessary to **provide the confidence** that a product conforms to a set of technical requirements
  - a set of activities designed to **evaluate the process** by which the software product is developed
- Galin
  - IEEE + respecting constraints on:
    - maintenance
    - time
    - money

# Quality factors...

- At the base of any attempt to produce good sw there is the need to have a good requirement documentation (if it is not known what needs to be done is difficult to get it right ...)
- In particular, in addition to the correct definition of functional and non-functional requirements, all the quality aspects essential for the application, such as:
  - usability
  - maintainability
  - reliability
  - ...

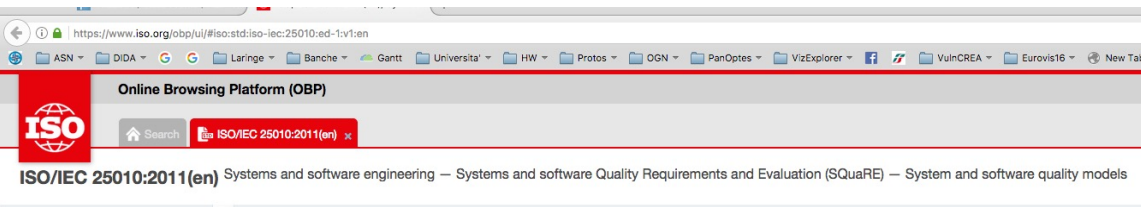
**MUST** be included in the requirements

# Product quality

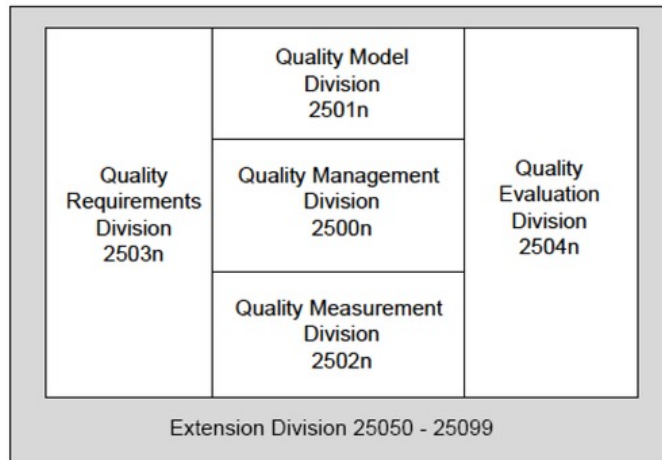
- It is obtained starting from:
  - **user's requirements**, which are specifications of quality as well as specified by the user, providing the first input to the design, and
  - **technical specifications**, which represent the quality required by the user translated by the developer in the software architecture, program structure, and the user interface

## ...Quality factors and standards

- That raised the need to classify what aspects of quality have to be included in requirements or, more generally, attributable to a software application (4 components)
- The first, and extremely actual proposal was from McCall (1977) and involved 11 quality factors
- The standard ISO/IEC 9126 Software engineering-Product quality, published for the first time in 1991 and revised and republished in 2001 builds up on the McCall and B. Boehm models
  - Software quality is defined as "*the set of characteristics that affect the ability of the product to satisfy **explicit** or **implicit** requirements.*" (definition very similar to that given in the ISO 9000 standard)
  - The software product is defined as "*the set of rules, procedures, programs, documents, relevant to the use of a computer system*"
- 9126 has been revised in 2011 as 25010



- **ISO/IEC 25010:2011** <https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en>
  - Systems and software engineering
  - **SQuaRE -- Systems and Software Quality Requirements and Evaluation**
  - System and software quality models
- [ISO/IEC 25010](#) cancels and replaces [ISO/IEC 9126-1:2001](#)
- [ISO/IEC 25010](#) is a part of the SQuaRE series of International Standards, which consists of the following divisions:
  - Quality Management Division (ISO/IEC 2500n)
  - Quality Model Division (ISO/IEC 2501n)
  - Quality Measurement Division (ISO/IEC 2502n)
  - Quality Requirements Division (ISO/IEC 2503n)
  - Quality Evaluation Division (ISO/IEC 2504n)
  - SQuaRE Extension Division (ISO/IEC 25050 – ISO/IEC 25099).



- ISO/IEC 2500n – **Quality Management Division**. The International Standards that form this division define all common models, terms and definitions further referred to by all other International Standards from the SQuaRE series. The division also provides requirements and guidance for a supporting function that is responsible for the management of the requirements, specification and evaluation of software product quality.
- ISO/IEC 2501n – **Quality Model Division**. The International Standards that form this division present detailed quality models for computer systems and software products, quality in use, and data. Practical guidance on the use of the quality models is also provided.
- ISO/IEC 2502n – **Quality Measurement Division**. The International Standards that form this division include a software product quality measurement reference model, mathematical definitions of quality measures, and practical guidance for their application. Examples are given of internal and external measures for software quality, and measures for quality in use. Quality Measure Elements (QME) forming foundations for these measures are defined and presented.
- ISO/IEC 2503n – **Quality Requirements Division**. The International Standards that form this division help specify quality requirements, based on quality models and quality measures. These quality requirements can be used in the process of quality requirements elicitation for a software product to be developed or as input for an evaluation process.
- ISO/IEC 2504n – **Quality Evaluation Division**. The International Standards that form this division provide requirements, recommendations and guidelines for software product evaluation, whether performed by evaluators, acquirers or developers. The support for documenting a measure as an Evaluation Module is also present.
- ISO/IEC 25050 – 25099 **SQuaRE Extension Division**. These International Standards currently include requirements for quality of Commercial Off-The-Shelf software and Common Industry Formats for usability reports.

# ISO/IEC 25010

- 25010 defines two **models**:
- A **quality in use** model composed of **5 characteristics** (some of which are further subdivided into subcharacteristics) that relate to the outcome of **interaction** when a product is used in a particular **context of use**. This system model is applicable to the complete human-computer system
- A **product quality** model composed of **8 characteristics** (which are further subdivided into subcharacteristics) that relate to **static** properties of software and **dynamic** properties of the computer system
- The characteristics defined by both models are relevant to all software products and computer systems. The subcharacteristics provide consistent terminology for **specifying, measuring and evaluating** system and software product quality. They also provide a set of quality characteristics against which stated **quality requirements** can be compared for completeness
- Old 9126 defined internal, external, and in use models



# Limits

- It is unlikely that the characteristics and sub-characteristic are always among them perfectly independent
- Characteristics and sub-characteristics are abstract properties that are related to one or more indicators that are measured by metrics. These are not always in linear relationship with characteristics they estimate
- It is missing, in any case, the link between the qualitative model and "how" to develop then good software

# 25010 objectives

- The scope of the models excludes purely functional properties, but it does include functional suitability
- Recipients are:
  - Users
  - Developers / maintainers
  - Client
  - System administrators
  - Customers

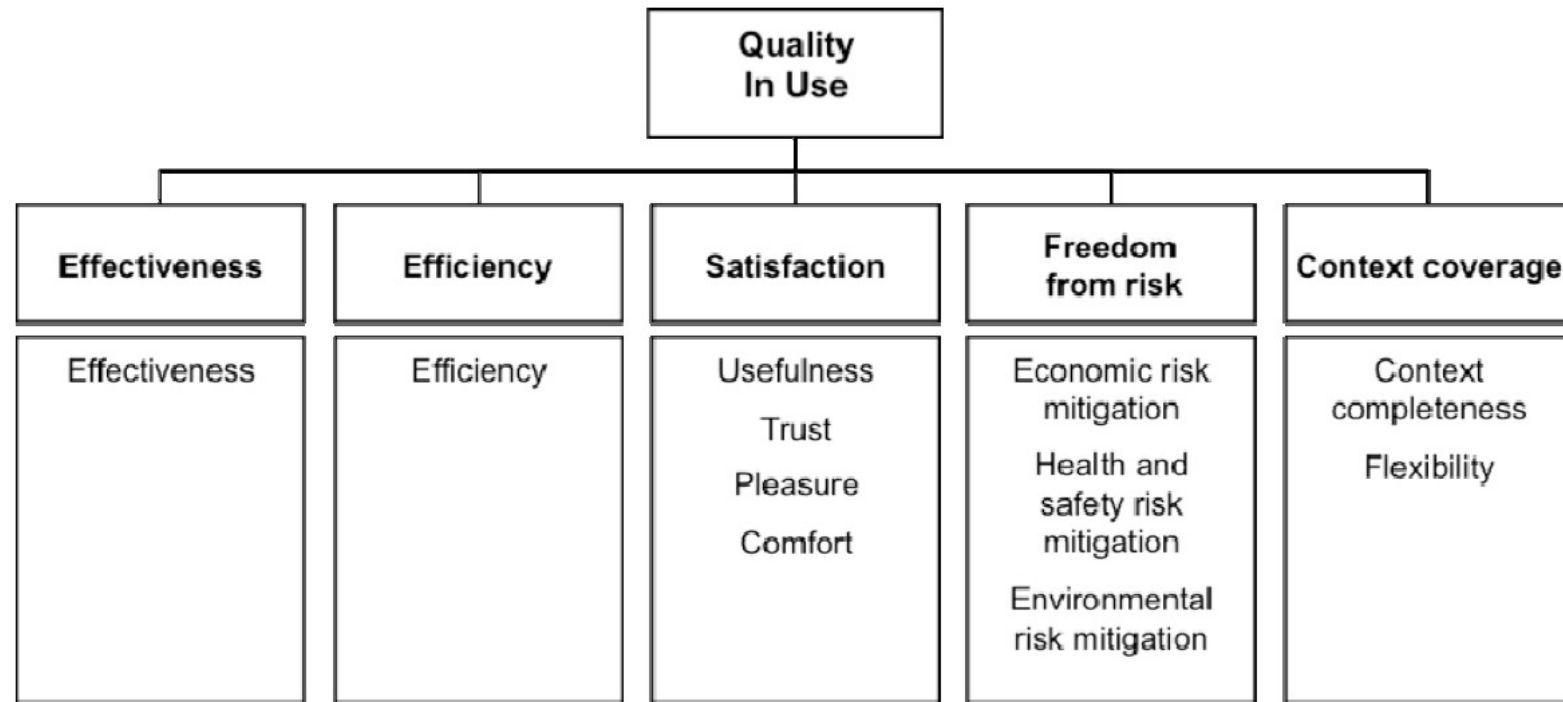
# 25010 at work

- Usage examples:
  - identifying software and system **quality** requirements
  - validating the **comprehensiveness** of requirements definition
  - identifying software and system **design objectives**
  - identifying software and system **testing objectives**
  - identifying **quality control criteria** as part of quality assurance
  - establishing **measures of quality characteristics** in support of these activities
  - identify the **criteria for acceptance** of products
  - provide a framework for the **definition of software quality** in a contractual document

# Quality in use

- Quality in use is what the user perceives in the use of the product in its actual context of use, in particular the ability of such a product to support him with effectiveness and efficiency in his work, exhibiting a good usability. This type of quality is what, above all, the developer must strive for

# In use quality model (5 characteristics)



## In use sw characteristics

- **Effectiveness**, the accuracy and completeness with which the user achieves specific goals
- **Efficiency**, the effort in relation to effectiveness
- **Satisfaction**, degree to which user needs are satisfied when a product or system is used in a specified context of use
  - **Usefulness**, degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use
  - **Trust**, degree to which a user or other stakeholder has confidence that a product or system will behave as intended
  - **Pleasure**, degree to which a user obtains pleasure from fulfilling their personal needs
  - **Comfort**, degree to which the user is satisfied with physical comfort

# In use sw characteristics

- **Freedom from risk**, degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment
  - **Economic risk mitigation**, degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use
  - **Health and safety risk mitigation**, degree to which a product or system mitigates the potential risk to people in the intended contexts of use
  - **Environmental risk mitigation**, degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use

# In use sw characteristics

- **Context coverage**, degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified
  - **Context completeness**, degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the **specified contexts** of use
  - **Flexibility**, degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in **contexts beyond** those initially specified in the requirements



# In use sw characteristics: METRICS?

## 1. EFFECTIVENESS

- $\# \text{ of reached objectives} / \# \text{ of target objectives}$  (proportion)
- $\# \text{ of correct reached objectives} / \# \text{ of reached objectives}$  (proportion)

## 2. Efficiency

- $\# \text{ of reached objectives} / \text{manpower}$  (ratio)

## 3. SATISFACTION

- Questionnaires (Likert scales)

## 4. Freedom for risk

- Safety incident rate (2 years)
- Normalized (KLOC)!

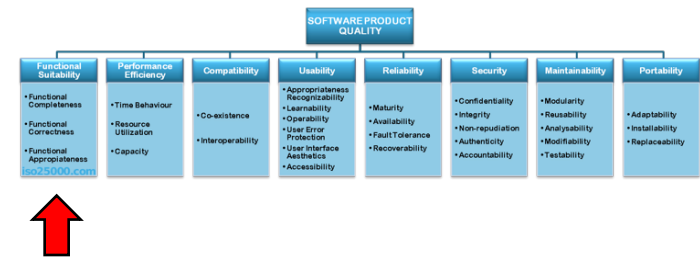
## 5. Context coverage

- $\# \text{ of successfully tested specified contexts} / \# \text{ of specified contexts}$  (proportion)
- $\# \text{ of successfully tested beyond contexts} / \# \text{ of specified contexts}$  (ratio)

# Product quality model (Static and dynamic)



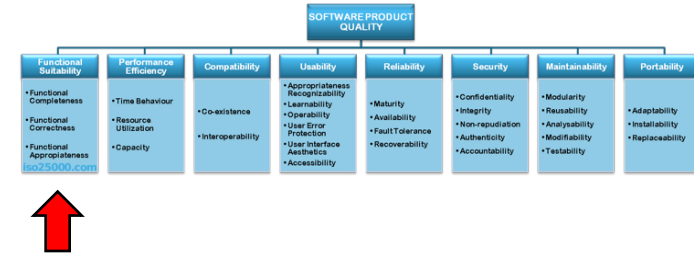
# Functional Suitability



**Functional suitability:** degree to which a product or system provides **functions** that **meet stated and implied needs** when used under specified conditions

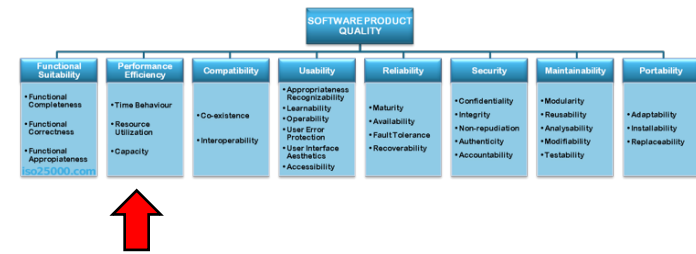
- functional completeness: degree to which the set of functions covers all the specified tasks and user objectives
- functional correctness: degree to which a product or system provides the correct results with the needed degree of precision
- functional appropriateness: degree to which the functions facilitate the accomplishment of specified tasks and objectives. **EXAMPLE:** A user is only presented with the necessary steps to complete a task, excluding any unnecessary step

# Functional suitability: metrics ?



- **Completeness:** completeness (to specification) of the functions of the software
  - # available functions / # required functions (proportion)
- **Correctness:** correctness of the functions (data come from tests)
  - #correct results / # results (proportion)
  - mean and standard deviation of the error (at least)
- **Appropriateness:** appropriateness (to specification) of the functions of the software
  - # appropriate functions / # functions (proportion)

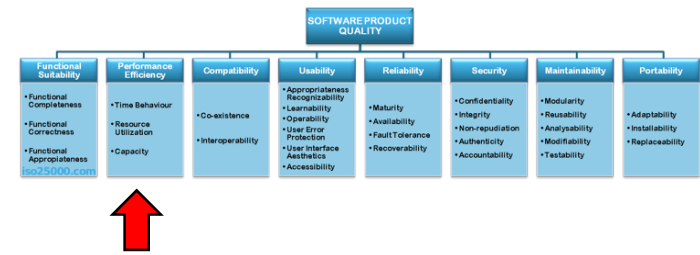
# Performance efficiency



**Performance efficiency:** performance relative to the amount of resources used under stated conditions (Resources can include other software products, the software and hardware configuration of the system, and materials (e.g., print paper, storage media))

- time behavior: degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements
- resource utilization: degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements (Human resources are included as part of efficiency)
- capacity: degree to which the maximum limits of a product or system parameter meet requirements (parameters can include the number of items that can be stored, the number of concurrent users, the communication bandwidth, throughput of transactions, and size of database)

# Performance efficiency : metrics



- **Time behavior**

- mean and standard deviation of the time needed to complete a function (max, min are useful as well)

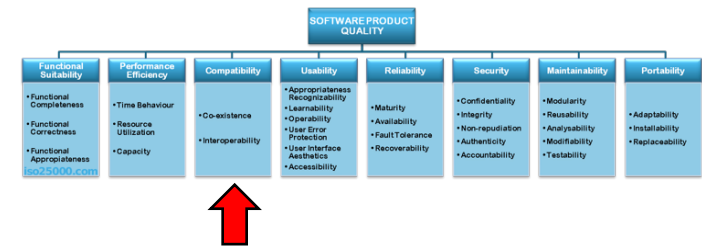
- main functions
    - all functions (weighted)

- **Resource utilization:**

- mean and standard deviation of

- CPU usage
    - Memory usage
    - Number of open file

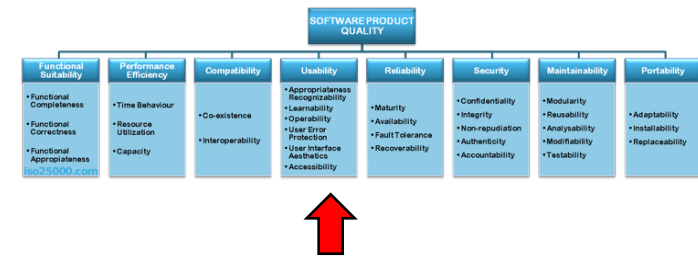
# Compatibility



**Compatibility:** degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment

- co-existence: degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product
- interoperability: degree to which two or more systems, products or components can exchange information and use the information that has been exchanged

# Usability

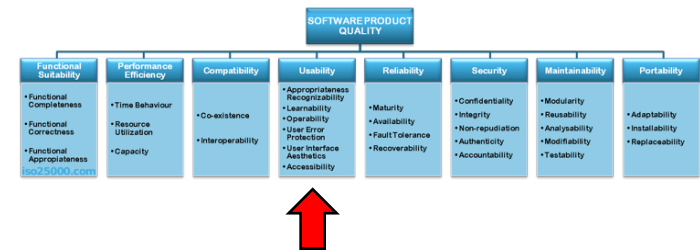


**Usability:** degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use (usability can either be specified or measured as a product quality characteristic in terms of its subcharacteristics, or specified or measured directly by measures that are a subset of quality in use)

- appropriateness recognizability: degree to which users can recognize whether a product or system is appropriate for their needs (appropriateness recognizability will depend on the ability to recognize the appropriateness of the product or system's functions from initial impressions of the product or system and/or any associated documentation)
- learnability: degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use



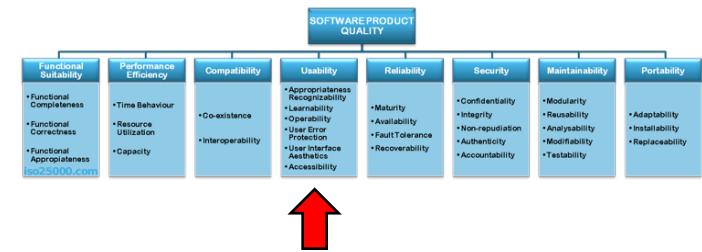
# Usability



## Usability:

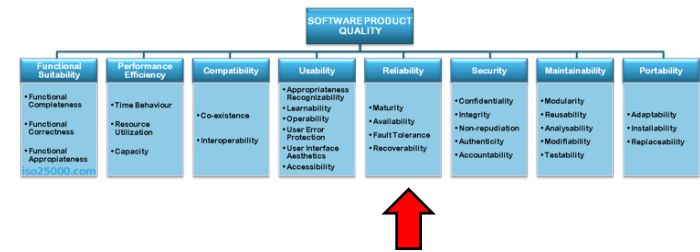
- operability: degree to which a product or system has attributes that make it easy to operate and control
- user error protection: degree to which a system protects users against making errors
- user interface aesthetics: degree to which a user interface enables pleasing and satisfying interaction for the user
- accessibility: degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use (The range of capabilities includes disabilities associated with age)

# USABILITY: Metrics ?



- **Appropriateness recognizability:**
  - mean and standard deviation of the time needed to **understand** the software functionalities
  - questionnaires (Likert scales)
- **Learnability :**
  - mean and standard deviation of the time needed to learn to **use** (up to 75% of the software functionalities)
  - questionnaires (Likert scales)
- **Operability:** easily operated
  - # of max three step functions / # of functions (proportion)
- **User interface aesthetics:**
  - questionnaires (Likert scales)

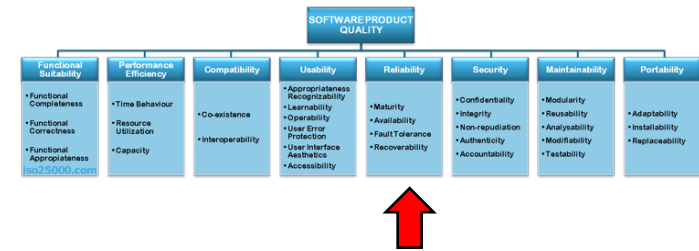
# Reliability



**Reliability:** degree to which a system, product or component performs specified functions under specified conditions for a specified period of time (Wear does not occur in software. Limitations in reliability are due to faults in requirements, design and implementation, or due to contextual changes)

- maturity: degree to which a system, product or component meets needs for reliability under normal operation
- availability: degree to which a system, product or component is operational and accessible when required for use (Externally, availability can be assessed by the proportion of total time during which the system, product or component is in an up state).
- fault tolerance: degree to which a system, product or component operates as intended despite the presence of hardware or software faults
- recoverability: degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system

# Reliability: metrics ?



- **Maturity**

- MTBF (mean time between failures)
- Software age
- Proportion between software developed for initial requirements and overall software, that includes less mature sw for fixing bugs, change request, etc.

- **Availability**

- Correct working time/usage time (real proportion: fraction)

- **Fault tolerance:**

- # of available functions after error  $e_i$  / # of available function (proportion)
- mean and standard deviation of available functions after a set of errors

- **Recoverability:**

- mean and standard deviation of the time needed to restore the system after an interruption
- mean and standard deviation of the data lost due to an interruption (proportion)

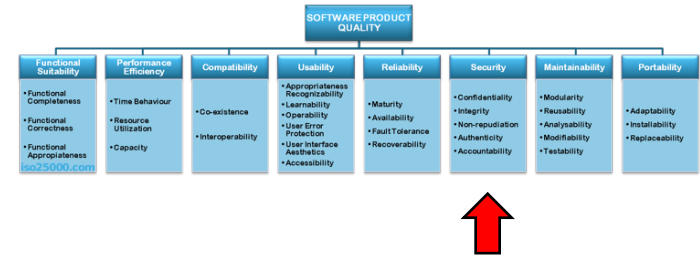
# Security



**Security:** degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization

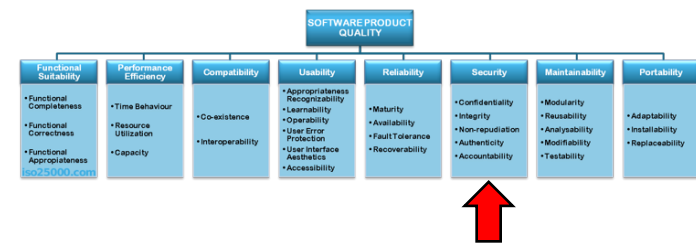
- confidentiality: degree to which a product or system ensures that data are accessible only to those authorized to have access
- integrity: degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data
- availability? CIA? not a Security sub-characteristic
- non-repudiation: degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later
- accountability: degree to which the actions of an entity can be traced uniquely to the entity
- authenticity: degree to which the identity of a subject or resource can be proved to be the one claimed

# Security: metrics?



- **Confidentiality/Integrity:** unauthorized access to the software functions
  - probability of non authorized access (difficult operative definition)
  - $\frac{\text{\# successful unauthorized access}}{\text{\# of unauthorized access}}$  (proportion)
  - length of the encryption key (indicator)

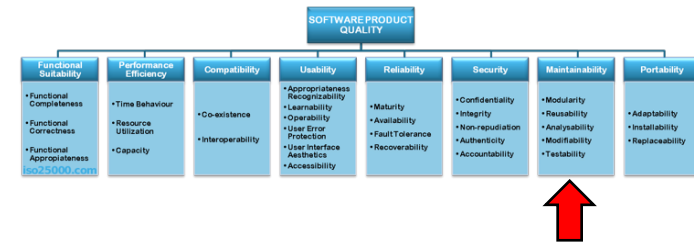
# Maintainability



**Maintainability:** degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers (Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications)

- modularity: degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components
- reusability: degree to which an asset can be used in more than one system, or in building other assets
- analysability: degree of effectiveness and efficiency with which it is possible to assess the **impact** on a product or system **of an intended change** to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified
- modifiability: degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality
- testability: degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met

# Maintainability : metrics?



- **Modularity:**
  - # of interclass calls / # of classes
- **Analysability :**
  - # errors sources discovered / manpower (ratio)
  - Mc Cabe cyclomatic complexity (indicator)
- **Modifiability:**
  - # fixed errors / manpower (ratio)
  - # new discovered errors during regression test / # fixed errors
- **Testability:**
  - # of tested changes / manpower (ratio)



# Portability

**Portability:** degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. Portability can be interpreted as either an inherent capability of the product or system to facilitate porting activities, or the quality in use experienced for the goal of porting the product or system.

- adaptability: degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments
- installability: degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment (If the product or system is to be installed by an end user, installability can affect the resulting functional appropriateness and operability)
- replaceability: degree to which a product can replace another specified software product for the same purpose in the same environment (e.g., replaceability of a new version of a software product is important to the user when upgrading)

# PORTABILITY: metrics

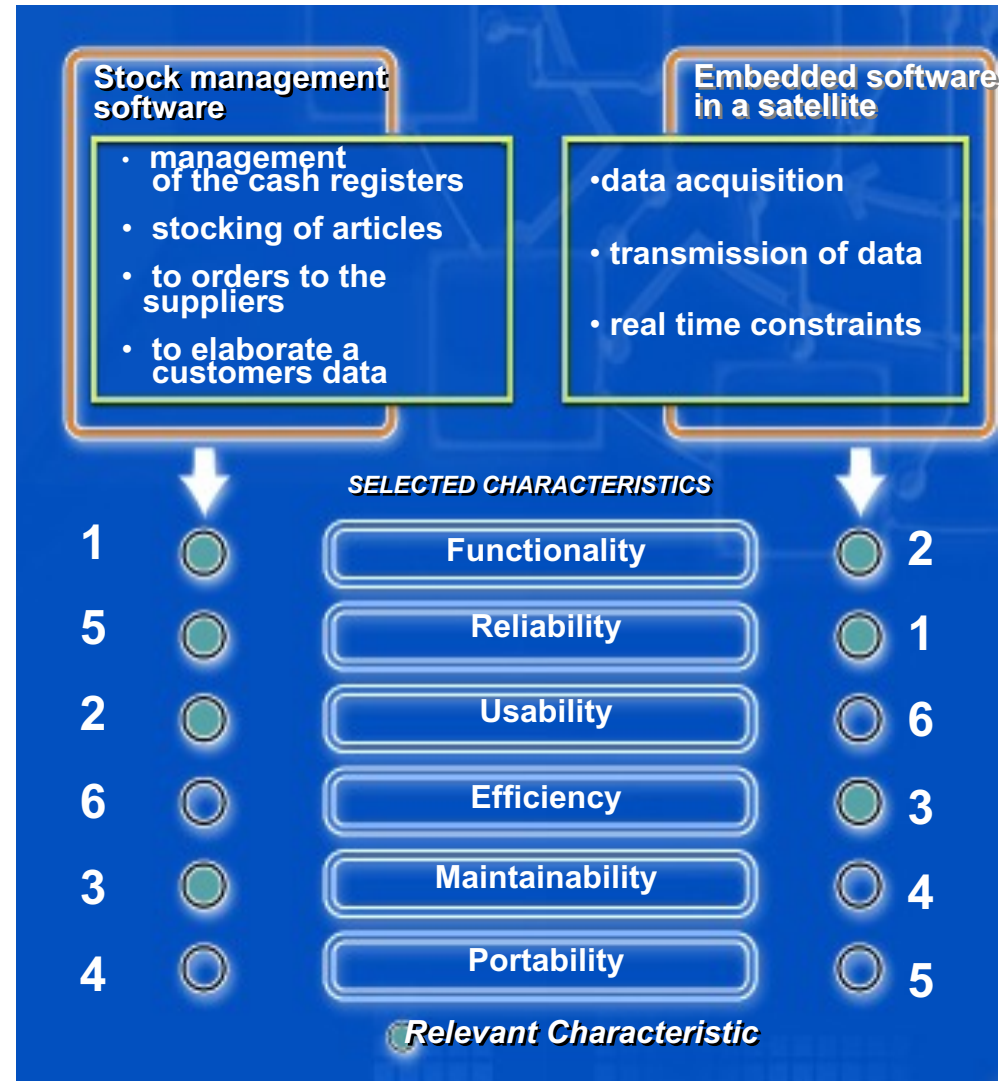


- **Adaptability :**
  - # of modifications after having changed the operating environment
  - manpower
  - normalized (KLOC /FP)
- **Installability :**
  - mean and standard deviation of the time needed to install the sw
  - # of problems / number of installations (ratio)
- **Replaceability :**
  - # of modifications
  - manpower
  - Training time
  - Normalized (KLOC /FP)

# Qualities are software specific

Quality features that a software must possess typically differ from product to product

A management software that organizes the activities of a warehouse, and a control software of a satellite, have very different quality requirements



# Measures

# 3 kinds of measures

1. **direct** - Detectable without the influence of external factors, such as the environment (hw and sw) in which the software "turns" the behavior and characteristics of users etc. .. Example of these measures are those found on the source code (static analysis and code reading) and those found by a reading of specific documents
2. **non-direct** - Derived from measures of one or more attributes. For example, measures relating to response times depend on not only the behavior of the software itself, but also the operating environment in which runs the software (hw and sw)
3. **indicators** – Some measures can be estimated from other measures (useful in the case of measures which cannot be detected directly). For example, the response time of software is not itself measurable when the sw is still in an unreachable state. So, the length of the code can be used as a simple and rough indicator of what will be the response time of the product into the environment of use

## 3 types of "semantics"

- Dimension
- Time
- Occurrences

# Dimension (absolute scale / ratio scale)

- Functional size, expressed through Function Points, but they can also be measured screenshots, files on which the application performs the calculations, etc.
- Program size, like:
  - **Logical Source stmt** (instructions), **Physical Source stmts** (LOC)
  - Program word count size, code vocabulary: operands and operators, (Halstead measures)
  - Program modules, classes
- Resource usage: disk space or memory, temporarily or permanently, % CPU, i/o and volume of data transferred, etc.
- ...

## Time (absolute scale / ratio scale)

- Real time, clock time, or running time
- Common measures:
  - System operation time, (e.g., reliability)
  - Execution time
  - User time (period spent by a user to complete a task )
  - Effort, time of production (for example, to develop x lines of code)
  - Time interval of events types, the time interval between two consecutive events in a period of observation (e.g., the interval between two consecutive component failures)



## Occurrences (nominal scale / absolute scale)

- the number of defects found in a test
- structural complexity measures, such as those of McCabe's
- number of inconsistencies with respect to a reference (e.g., non-compliance with respect to a rule or a standard encoding)
- the number of changes to a program (or the number of LOC/FP/STMT modified)
- the number types of defects detected (according to a classification)
- the number of shares necessary to interact with a computer to perform a task, (e.g., the number of keys to press, the number of movements of the eye needed etc.)
- ...

# Measure compositions

- Occurrences (Count), Dimension (Size) and Time (Time)
- 9 possible combinations

# Composition (1) : dividing by time

- Rates
- Frequencies
- Time proportions

<i>Composition</i>	<i>Suitable for</i>	<i>Examples</i>
<b>Count / Time</b>	<b>Rates Frequencies</b>	<ul style="list-style-type: none"><li>– User mistake rate in a month</li><li>– Defect rate in the first two usage years</li><li>– Transactions per second</li></ul>
<b>Size / Time</b>	<b>Frequencies</b>	<ul style="list-style-type: none"><li>– Software LOC developed in one day</li></ul>
<b>Time / Time</b>	<b>Time proportions</b>	<ul style="list-style-type: none"><li>– % of operative time</li></ul>
		–

# Dividing by occurrences

- Occurrence proportion
- Mean
- Time interval

<i>Combination</i>	<i>Suitable for</i>	<i>Examples</i>
Count / Count	Occurrence proportions Occurrence mean	<ul style="list-style-type: none"><li>– % of test cases that discovered at least one error</li><li>– mean of discovered errors</li></ul>
Size / Count	size means	<ul style="list-style-type: none"><li>– mean of LOC per method</li></ul>
Time / Count	intervals time means	<ul style="list-style-type: none"><li>– Time between two errors (MTBF)</li><li>– Mean of the time required to fix a defect</li></ul>

# Dividing by size

- Densities
- Dimension proportions
- Efficiency / Effort

<i><b>Combination</b></i>	<i><b>Suitable for</b></i>	<i><b>Examples</b></i>
<b>Count / Size</b>	<b>Parameter densities</b>	<b>– Defect density</b>
<b>Size / Size</b>	<b>Dimension proportions</b>	<b>– % of implemented FP with respect to the planned FP</b>
<b>Time / Size</b>	<b>Efficiency / Effort</b>	<b>– Execution time of a LOC</b> <b>– Required effort to test a LOC</b>

# Visualizations

The measurements must be displayed

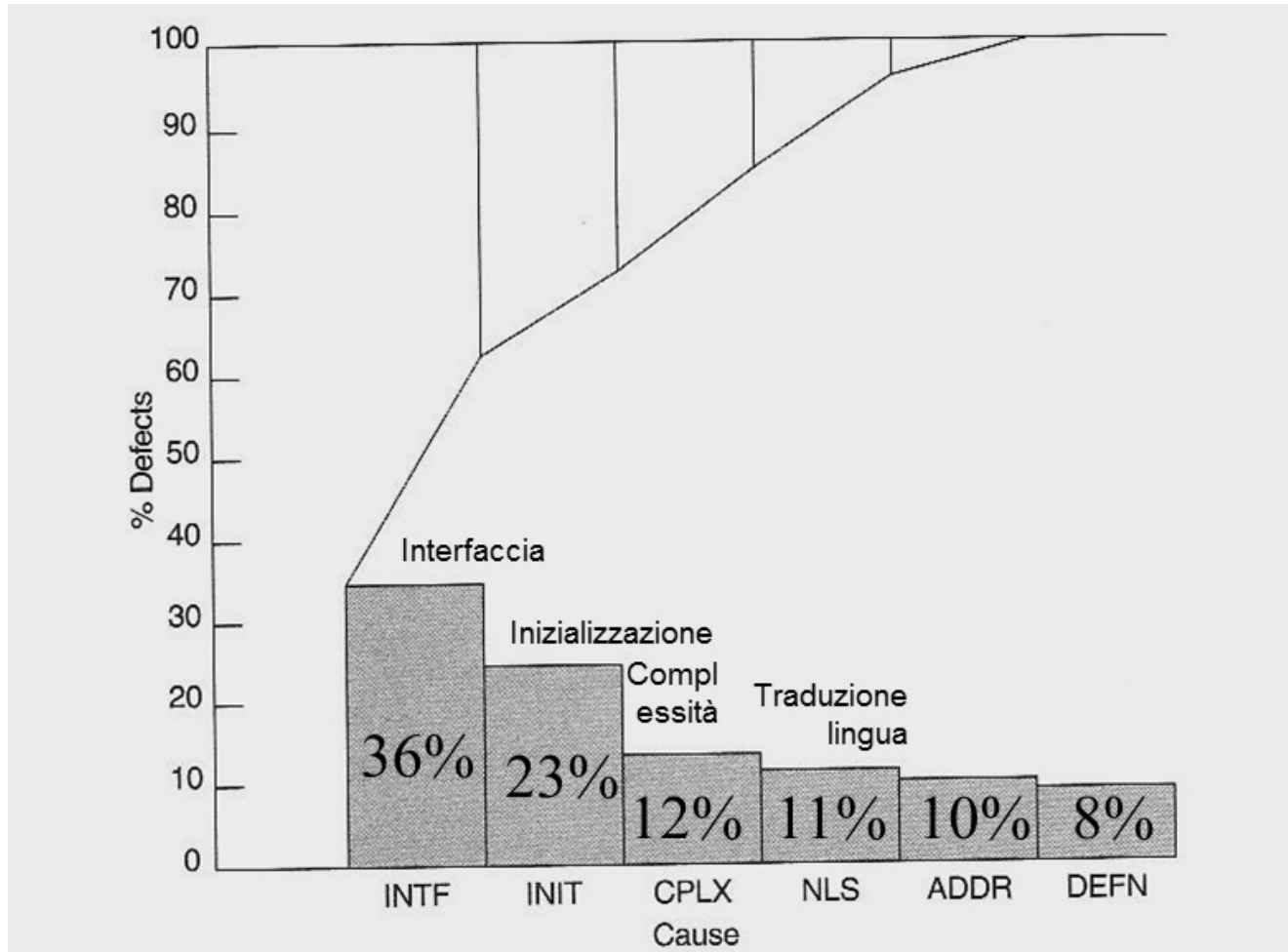
The ISO recommends these views:

frequency vs. measures (histograms, Pareto diagrams)

time vs. measures (trend analysis or predictive estimates)

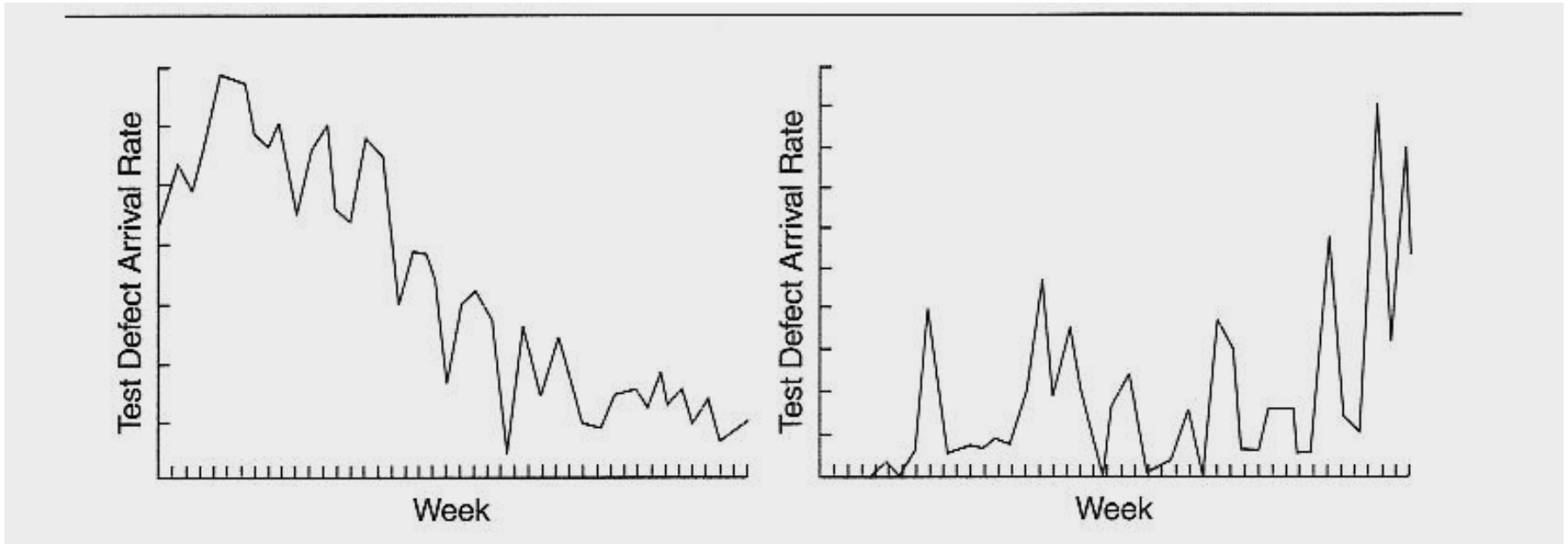
measures vs. measures (scatter plot, correlation, etc.)

# Pareto diagrams



## Defect distribution

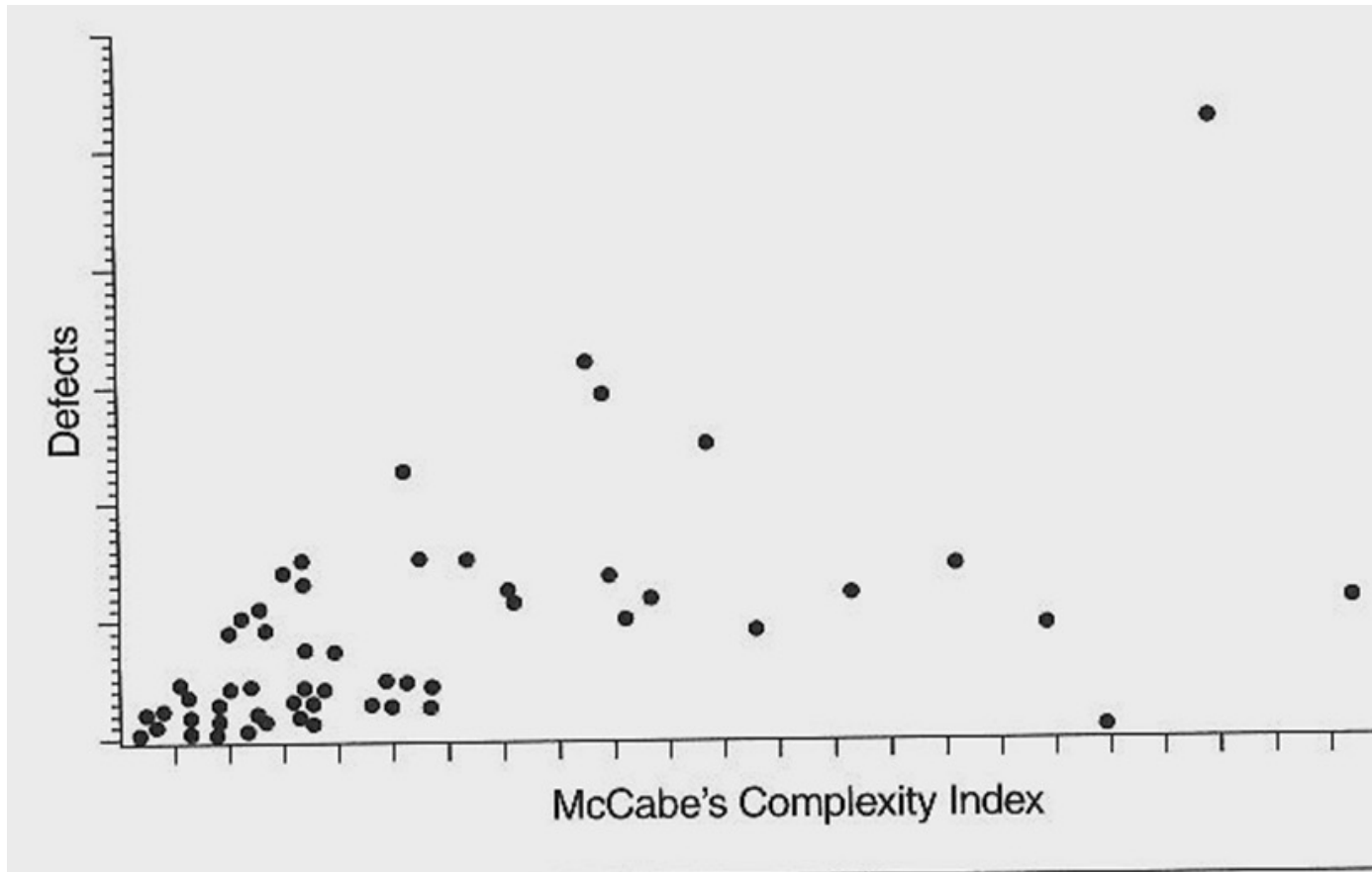
# Time



## Test defect arrival rate



# Scatter plot



Looking for a correlation

# Examples of static metrics – Functional suitability

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric Scale Type	Measure type	Source of Input to measurement	12207 reference	Beneficiaries
Functional appropriateness	How appropriate are the checked functions?	Measure the ratio of implemented functions that are suitable for performing the specified tasks to those implemented. The following may be measured; -all or parts of design specifications -completed modules/parts of software products	$X=1-A/B$ A= Number of functions in which problems are detected in evaluation B= Number of functions checked	$0 \leq X \leq 1$ The closer to 1, the more adequate.	absolute	X=count/ count A=count B=count	Req spec Design Source code  Review report	6.5Validation 6.6Joint review	Requirer Developer
Functional implementation completeness	How complete is the functional implementation?	Count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications  1.	$X=1-A/B$ A=Number of missing functions detected in evaluation. B=Number of functions described in requirement specifications  NOTE: Input to the measurement process is the updated requirement specifications. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.	$0 \leq X \leq 1$ The closer to 1, the more completed.	absolute	X=count/ count A=count B=count	Req spec Design Source code  Review report	6.5Validation 6.6Joint review	Requirer Developer

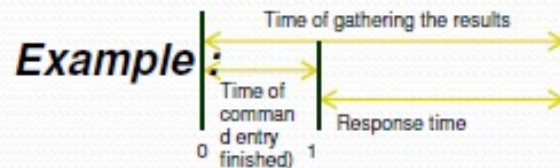
## Examples of dynamic metrics – Efficiency

**Characteristics: Efficiency**

**Sub-characteristic: Time behaviour**

**Example of measure: Response time**

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	ISO/IEC 12207 SLCP Reference	Target audience
<b>Response time</b>	What is the time taken to complete a specified task?	Start a specified task. Measure the time it takes for the sample to complete its operation.	$T = (\text{time of gaining the result}) - (\text{time of command entry finished})$	$0 < T$ The sooner is the better.	Ratio	T= Time	Testing report	5.3 Sys./Sw.	User
	How long does it take before the system response to a specified operation?	Keep a record of each attempt.					Operation report showing elapse time	5.3 Integration testing 5.4 Qualification testing 5.5 Maintenance	Developer Maintainer SQA



**Practical usage**  
**AIPA recommendations for**  
**Public administration software**  
**(based on 9126)**

**(Autorità per l'Informatica nella Pubblica Amministrazione)**

# Functionality (1)

°	Sub-char	Indicatore	Formula	Tecnica di misura	Valore soglia
	Adeguatezza (suitability)	copertura funzionale	$\frac{\text{n° di funzioni presenti nel prodotto}}{\text{numero di funzioni specificate nel progetto esecutivo}}$	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%
		copertura documentazione	$\frac{\text{n° di funzioni descritte nel manuale}}{\text{numero di funzioni presenti nel prodotto}}$	checklist applicate alla documentazione	100%
		copertura test	$\frac{\text{n° di test eseguiti}}{\text{numero di test specificati nel progetto esecutivo}}$	checklist applicate alla documentazione, verifiche ispettive	99%
	Accuratezza (accuracy)	accuratezza documentazione	$\frac{\text{n° di funzioni esattamente descritte nel manuale}}{\text{numero di funzioni descritte nel manuale}}$	checklist applicate alla documentazione	80%

## Functionality (2)

Interoperabilità	tasso di interfacce incontrate	n° di interfacce realizzate / n° di interfacce specificate nel progetto esecutivo	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%
Aderenza (compliance)	tasso di interfacce standardizzate rispetto a quelle esistenti in azienda	n° di interfacce standardizzate / n° di interfacce che devono essere standardizzate secondo il progetto esecutivo	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%
Sicurezza (security)	copertura sicurezza informazioni	n° di informazioni riservate che dispongono di “log” di accesso / n° di informazioni riservate specificate nel progetto esecutivo	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%
	copertura sicurezza informazioni	n° di informazioni riservate con accesso limitato / n° di informazioni riservate con accesso limitato specificate nel progetto esecutivo	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%

# Reliability

°	Sub-char	Indicatore	Formula	Tecnica di misura	Valore soglia
	Maturità (maturity)	densità di errori sul prodotto (dopo i test finali)	$\frac{\text{n° degli errori applicativi}}{\text{volume del prodotto operativo}}$	Verifiche su archivi di log ed interviste all'utenza	0.1 (in un anno/FP)
	Tolleranza (fault tollerance)	tasso di errori applicativi che hanno provocato un fermo della applicazione	$\frac{\text{n° degli errori applicativi}}{\text{n° degli errori applicativi che hanno provocato dei fermi}}$	Verifiche su archivi di log	1%
	Ripristinabilità (recoverability)	tasso di disponibilità	$\frac{\text{totale del tempo operativo}}{\text{totale del tempo di osservazione}}$	Verifiche su archivi di log	98%
		tempo medio di vita di un errore	$\frac{\text{tempo totale di vita di un errore}}{\text{n° degli errori osservati}}$	Verifiche su archivi di log	72 ore

Usability(1)

°	Sub-char	Indicatore	Formula	Tecnica di misura	Valore soglia
	Facilità di comprensione (understandability )	Facilità di comprensione del manuale	Tempo necessario ad un operatore medio a richiedere una funzione (guidata dal manuale)	Verifica ispettiva (Walkthroughs)	entro 10 minuti qualsiasi funzione
	Apprendibilità	Disponibilità di help in linea	n°. di help / n° di oggetti (funzioni/campi)	checklist applicate alla documentazione e verifiche ispettive sul prodotto	50%
		Disponibilità di funzioni di apprendimento in linea	n° di funzioni di “learning” / n° di oggetti (funzioni/campi)	checklist applicate alla documentazione e verifiche ispettive sul prodotto	50%



## Usability (2)

Operabilità	Grado di disponibilità di valori di default	n° di comandi e campi dei menu che dispongono di valori di default / Totale dei comandi e dei campi che li ammettono secondo il progetto esecutivo	checklist applicate alla documentazione e verifiche ispettive sul prodotto	90%
	Grado di disponibilità di liste di dati su cui scegliere per dare input alle applicazioni	n° di liste disponibili / totale dei campi dove sarebbero applicabili	checklist applicate alla documentazione e verifiche ispettive sul prodotto	90%
	Uniformità dei comandi	n° dei comandi che hanno formato standard / totale n° dei comandi	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%
	n° medio di tasti da premere per eseguire una funzione (o di click sul mouse)	n° di tasti premuti per eseguire una funzione	verifiche ispettive	max 10
	Intervallo tra due errori umani nell'utilizzo della applicazione	Intervallo medio temporale tra due successivi errori umani	verifiche ispettive	10 minuti

# Efficiency

°	Sub-char	Indicatore	Formula	Tecnica di misura	Valore soglia
		Response time	Intervallo di tempo medio tra immissione di una richiesta a terminale che non richiede trattamento di dati, ed ottenimento della risposta a terminale	Verifiche su archivi di log e verifiche ispettive sul prodotto (e sul codice)	10 sec
		Turnaround time	Intervallo di tempo medio tra immissione di una richiesta che richiede trattamento di dati ed ottenimento dell'output relativo	Verifiche su archivi di log e verifiche ispettive sul prodotto (e sul codice)	TP=10sec Stampa=2m

# Portability

°	Sub-char	Indicatore	Formula	Tecnica di misura	Valore soglia
	Adattabilità	Tasso di esportabilità del patrimonio dati	$\frac{\text{n}^\circ. \text{ di dati esportabili}}{\text{n}^\circ. \text{ di dati}}$	checklist applicate alla documentazione e verifiche ispettive sul prodotto	100%
		Tasso di esportabilità dell'ambiente operativo	$\frac{\text{n}^\circ. \text{ di funzioni esportabili}}{\text{n}^\circ. \text{ di funzioni}}$	checklist applicate alla documentazione e verifiche ispettive sul prodotto	90%
		Tasso di modifica parametri per cambiamento ambiente operativo	$\frac{\text{n}^\circ. \text{ di parametri da modificare}}{\text{n}^\circ. \text{ di parametri}}$	checklist applicate alla documentazione e verifiche ispettive sul prodotto	50%
		Tasso di ricompilazione per cambiamento ambiente operativo	$\frac{\text{n}^\circ. \text{ di programmi da ricompilare}}{\text{n}^\circ. \text{ di programmi}}$	checklist applicate alla documentazione e verifiche ispettive sul prodotto	50%

# So what?

- How to enforce the software quality:



## Software Quality Assurance (SQA)

<http://www.sqa.net/index.htm>

Nasa definition:

**The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented**

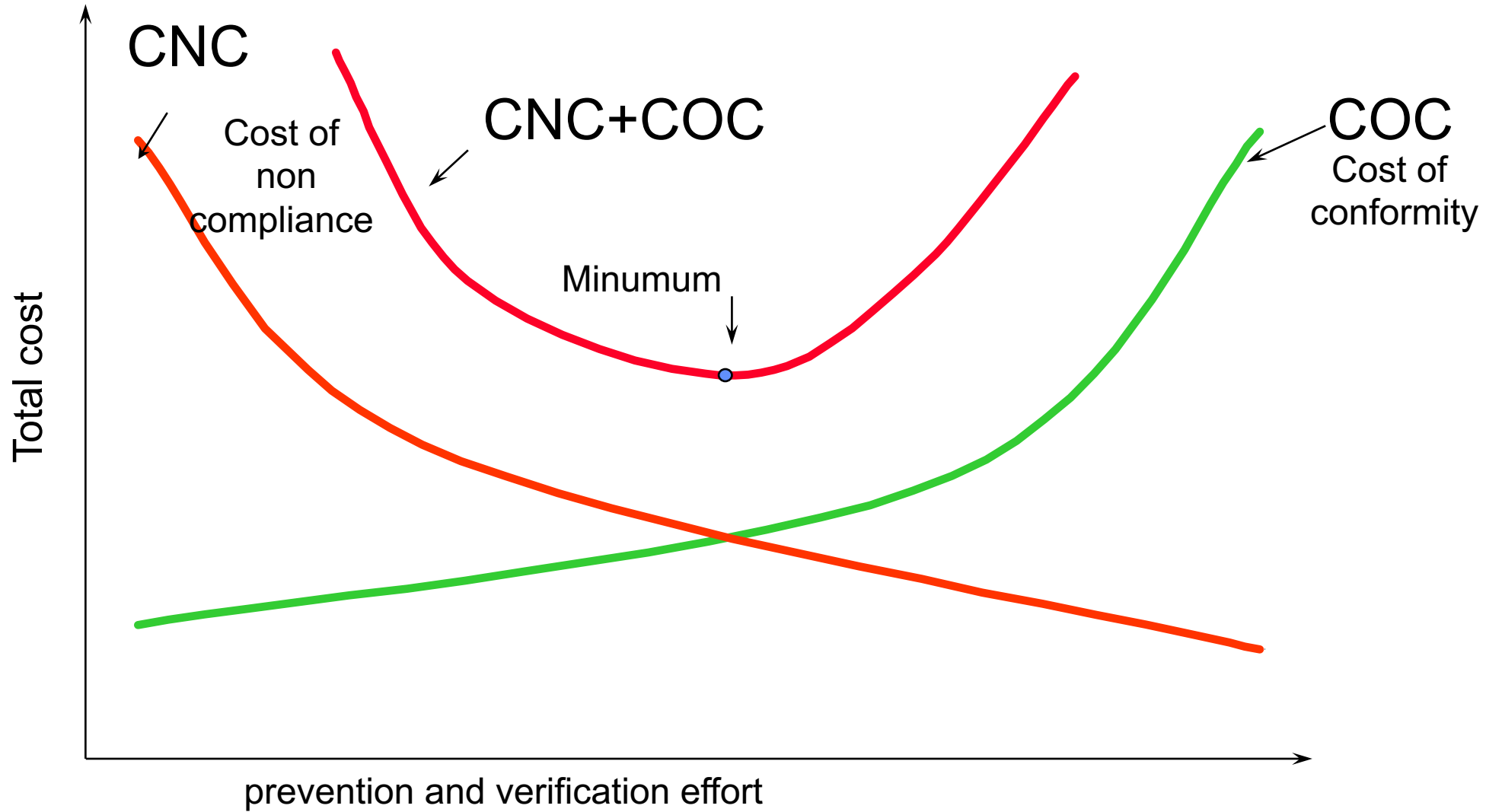
# Quality cost

- Due expense sources:
  - To ensure quality
  - To fix errors




# Quality cost (ISO)

- The costs of quality in the production process are the costs that you bear to adapt the quality of the product to the required quality
- **Cost of conformity (COC)** (to satisfy all the needs expressed and implied)
  - Prevention costs: costs incurred to prevent failures cost assessment:
  - Inspection cost : costs for inspections and tests
- **Cost of non-compliance (NC)** (for internal and external failures )
  - costs for internal failures: charges related to a product that does not meet the quality requirements before its delivery
  - costs for external failures: charges related to a product that does not meet the requirements of quality after its delivery (maintenance costs and repair, warranty costs, costs and returns for the recall of the products, costs for product liability, etc.)

# Quality cost



# ISO and other organizations

		<u>General</u>	<u>Electrotechnical</u>
<u>International level</u>		<i>ISO</i>	<i>IEC</i>
<u>European level</u>		<i>CEN</i>	<i>CENELEC</i>
<u>National level</u>		<i>UNI</i>	<i>CEI</i>



# Acronyms

- ISO International Organization for Standardization
  - IEC International Electrotechnical Commission
  - CEN Comitato Europeo di Normazione (sigla sui documenti EN)
  - CENELEC Comitato Europeo di Normazione Elettrotecnica (sigla sui documenti EN HD)
  - UNI Ente Nazionale Italiano di Unificazione
  - CEI Comitato Elettrotecnico Italiano
- 
- Non European National ISO:
    - ANSI American National Standard Institute, United States;
    - JISC Japan Industrial Standards Committee, Japan;
    - SA Standards Australia, Australia;
    - SCC Standard Council of Canada, Canada.

# Standard evolution

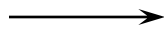
PROCESS

PRODUCT

CMM (SEI 87)

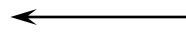


ISO/IEC 12207 (91)



ISO

14598



ISO/IEC 9126 (91)

ISO 9001 - UNI-EN 29001 (88)

revised on 2000: Vision 2000  
revised on 2015: ISO 9001:2015

CMMI

ISO/IEC 9126 (2001)

ISO 25010:2011

# Exercise

- A little city is installing some self service systems to buy train tickets. The system has an interface through touch-sensitive screen and allows the user to specify the trip and to pay cash or with a credit card. Indicate two quality requirements that are relevant to the application, accompanying each one with a metric:
  - measuring procedure
  - scale
  - type of measure
  - internal/external/in use
  - direct, indirect, indicator
  - size, time, occurrence and their possible combination

# Usability

- **Learnability** Learning effort for different users, i.e. novice, expert, casual etc.
- **Metric:** mean/ standard deviation time for completing a transaction (threshold? Mean about a minute ?)
  - measuring procedure: Log analysis
  - scale: ratio scale
  - in use
  - indicator
  - time/count

# Usability

- **Operability** Ability of the software to be easily operated by a given user in a given environment
- **Metric** : mean/ standard deviation time for completing each subtask (selecting stations, time, payment)
  - measuring procedure: Log analysis
  - scale: ratio scale
  - In use
  - indicator
  - time/count

# Security

- **Confidentiality :**
  - degree to which a product or system ensures that data are accessible only to those authorized to have access
- **Metric :** number of stolen credit card numbers/year
  - measuring procedure: Log analysis(?)
  - police reports?
  - scale: rate
  - in use
  - indicator
  - count/time