

# Measurement

---

## Empirical Methods in Software Engineering (01 OPJIU)



<http://softeng.polito.it/EMSE/>  
**SoftEng**  
<http://softeng.polito.it>

Version 1.5.1  
© Marco Torchiano, 2017






This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-nc-nd/4.0/>.

You are free: to copy, distribute, display, and perform the work

Under the following conditions:

-  **Attribution.** You must attribute the work in the manner specified by the author or licensor.
-  **Non-commercial.** You may not use this work for commercial purposes.
-  **No Derivative Works.** You may not alter, transform, or build upon this work.
  - For any reuse or distribution, you must make clear to others the license terms of this work.
  - Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

# Contents

---

- Introduction to measurement
- Theory of measurement
- Measurement scales
- Software Product Quality measures
- Common metrics

# Measurement

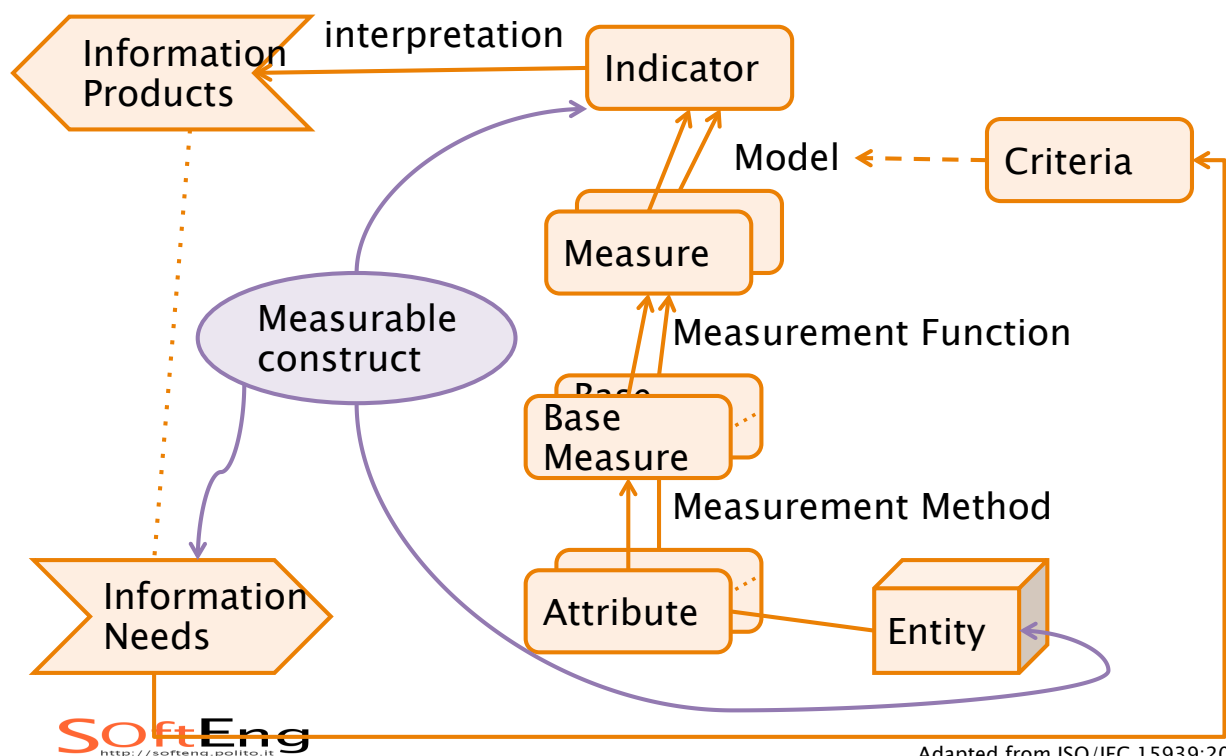
---

*the process of empirical  
**objective** assignment of  
numbers to **entities**, in order  
to characterize a specific  
**attribute** thereof*

# Measurement

- Entity:
  - ♦ an object or event
- Attribute:
  - ♦ a feature or property of an entity
- Objective:
  - ♦ the measurement process must be based on well-defined rules and procedures whose results are repeatable

## Measurement Conceptual Model



# Terms

---

**Measure** (noun): variable to which a value is assigned as the result of measurement.

**Measure** (verb): Make a measurement.

**Measurement**: The process of assigning a number or category to an entity to describe an attribute of that entity.

**Metric**: A measurement scale and the method used for measurement

**Indicator**: Measure that provides an estimate or evaluation derived from a model with respect to defined information needs

## Examples of metrics

Entity	Attribute	Measure
Person	Age	Year of last birthday
Person	Age	Months since birth
Source code	Length	# Lines of Code (LOC)
Source code	Length	# Executable statements
Testing process	Duration	Time in hours from start to finish
Tester	Efficiency	Number of faults found per KLOC
Testing process	Fault frequency	Number of faults found per KLOC
Source code	Quality	Number of faults found per KLOC
Operating system	Reliability	Mean Time to Failure

# Guidelines

---

- Specify both entity and attribute
  - ♦ The entity must be defined precisely
- You must have a reasonable, even just intuitive understanding of the attribute before you propose a measure.
- You must not re-define an attribute to fit in with an existing measure.

# Common mistake

---

- Mistake: propose a ‘measure’ if there is no consensus on what attribute it characterizes.
- Results of an IQ test
  - ♦ Intelligence?
  - ♦ or verbal ability?
  - ♦ or problem solving skills?
- # defects found / KLOC
  - ♦ quality of code?
  - ♦ quality of testing?

# Type and use

---

- Types
  - ◆ Direct measurement
  - ◆ Indirect measurement
- Uses of measurement:
  - ◆ for assessment
  - ◆ for prediction
    - Measurement for prediction requires a prediction system/model

## Direct measures

---

- **Length** of source code
  - ◆ E.g. measured by LOC
- **Duration** of testing process
  - ◆ E.g. measured by elapsed time in hours
- **Number of defects** discovered during the testing process
  - ◆ E.g. measured by counting defects
- **Effort** of a programmer on a project
  - ◆ E.g. measured by person months worked

# Indirect measures

---

$$\text{Programmer productivity} = \frac{\text{LOC produced person}}{\text{months of effort}}$$

$$\text{Module defect density} = \frac{\text{number of defects module}}{\text{size}}$$

$$\text{Defect detection efficiency} = \frac{\text{number of defects detected}}{\text{total number of defects}}$$

$$\text{Requirements stability} = \frac{\text{\# of initial requirements}}{\text{total \#of requirements}}$$

$$\text{Test effectiveness ratio} = \frac{\text{number of items covered}}{\text{total number of items}}$$

$$\text{System spoilage} = \frac{\text{effort spent fixing faults}}{\text{total project effort}}$$

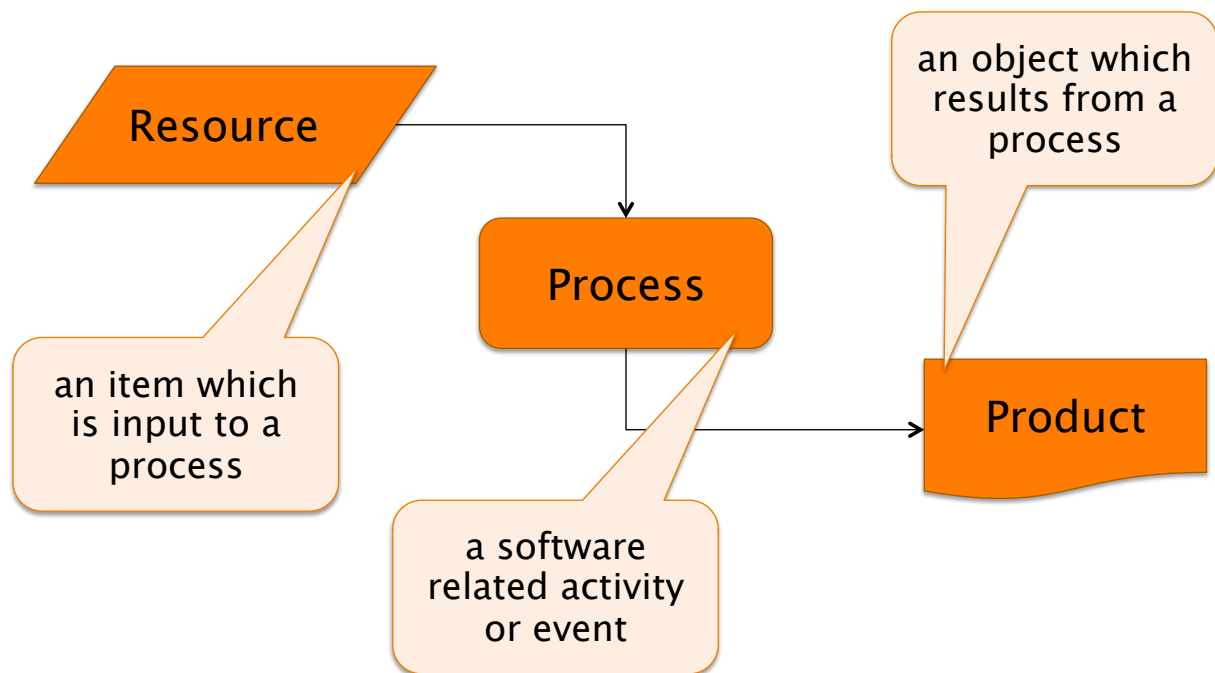
# Predictive measurement

---

- Requires a *prediction system*
  - ♦ Mathematical model
    - e.g. ‘ $E=as^b$ ’ where  $E$  is effort in person months (to be predicted),  $S$  is size (LOC), and  $a$  and  $b$  are constants.
  - ♦ Procedures for determining model parameters
    - e.g. ‘Use regression analysis on past project data to determine  $a$  and  $b$ ’.
  - ♦ Procedures for interpreting the results
    - e.g. ‘Use Bayesian probability to determine the likelihood that your prediction is accurate to within 10%’

# Entity classes

---



## Internal vs. External

---

Given an entity (process, product, or resource)

- **Internal** attributes can be measured purely in terms of the entity itself (static)
  - ♦ e.g. length or complexity of source code (product)
- **External** attributes can only be measured with respect to how the entity relates to its environment (dynamic)
  - ♦ e.g. reliability or maintainability of source code (product)

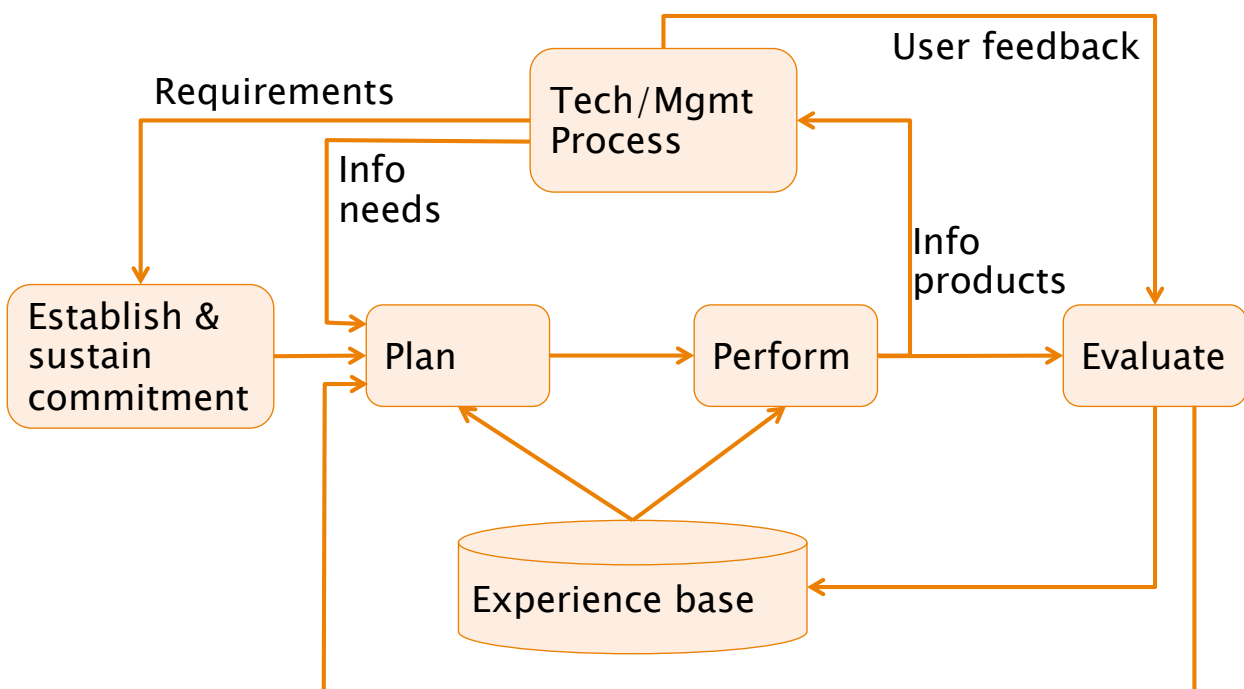


# Metrics

Entities	Attributes	
	Internal	External
<b>PRODUCTS</b> <i>Specification</i> <i>Source Code</i> ....	Length, functionality modularity, structuredness, reuse ....	maintainability reliability .....
<b>PROCESSES</b> <i>Design</i> <i>Test</i>	time, effort, #spec faults found ....	stability cost- effectiveness .. ..
<b>RESOURCES</b> <i>People, Tools</i> ....	age, price, CMM level price, size ....	productivity usability, quality ....

17

# Measurement Process



18

# Warning

---

*The more any quantitative social indicator is used for social decision-making, the more subject it will be to corruption pressures and the more apt it will be to distort and corrupt the social processes it is intended to monitor.*

- ◆ Campbell's law

---

## MEASUREMENT THEORY BASICS

# Evolution of metrics

---

- Metrics depend on the understanding of the attribute
- More sophistication as understanding of an attribute grows
- E.g. temperature of matter:
  - ◆ 200BC: rankings, “hotter than”
  - ◆ 1600: first thermometer still “hotter than”
  - ◆ 1720: Fahrenheit scale
  - ◆ 1742: Centigrade scale
  - ◆ 1854: Absolute zero, Kelvin scale

# Measurement theory

---

- Scientific basis to determine formally:
  - ◆ When we have really defined a measure
  - ◆ Which statements involving measurement are meaningful
  - ◆ What the appropriate scale type is
  - ◆ What types of statistical operations can be applied to measurement data
- Based on foundation laid down by Stevens (1946)

# Empirical relation system

---

- A set of **entities**
- The **relations** about entities, observed in the real world, which characterize our understanding of the attribute under consideration
  - ♦ e.g. 'Fred taller than Joe' (for *height* of *people*)
- The closed **operations** that can be performed on the objects

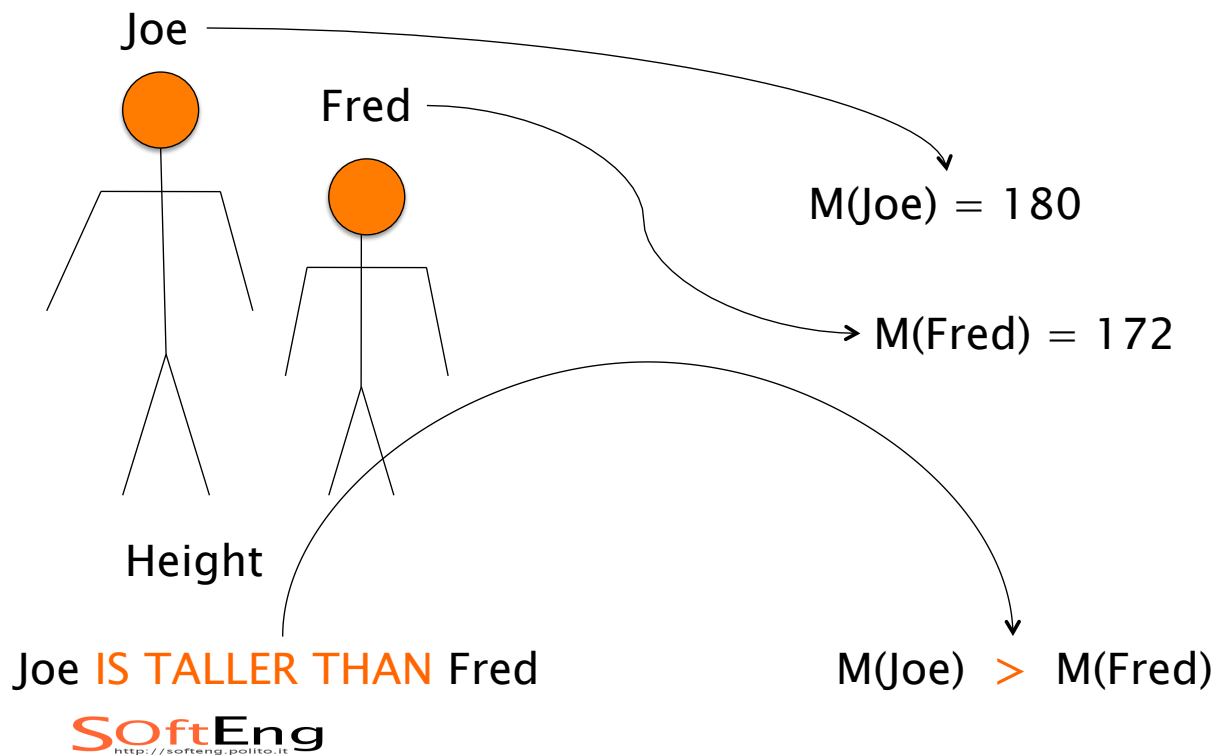
# Measurement mapping

---

- Mapping from the empirical world to the formal world
  - ♦ Measure
  - ♦ Relation mapping
- A.k.a. representation, homomorphism
- Measure: the quantity assigned to an entity in order to characterize an attribute

# Measurement mapping

---



25

# Representation condition

---

- Measurement mapping implies that all empirical relations are preserved in numerical relations and no new relations are created
  - ◆ e.g.  $M(\text{Fred}) > M(\text{Joe})$  precisely when Fred is taller than Joe
- **Admissible measure** if the representation condition holds
  - ◆ Measurement **scale**

26

# Formally

---

- We can define a homomorphism  $m$

scale:  $(\mathcal{E}, \mathfrak{F}, m)$   
empirical system:  $\mathcal{E} = (E, \text{taller})$   
formal system:  $\mathfrak{F} = (\mathbb{R}, >)$   
mapping function:  $m : E \rightarrow \mathbb{R}$   
 $\forall a, b \in E, a \text{ taller } b \implies m(a) > m(b)$

# Additive metric

---

- A possible additional requirement is to have an additive measure

scale:  $(\mathcal{E}, \mathfrak{F}, m)$   
empirical system:  $\mathcal{E} = (E, \text{taller, added})$   
formal system:  $\mathfrak{F} = (\mathbb{R}, >, +)$   
mapping function:  $m : E \rightarrow \mathbb{R}$   
 $\forall a, b \in E :$   
 $a \text{ taller } b \implies m(a) > m(b)$   
 $m(a \text{ added } b) = m(a) + m(b)$

# Admissible transformation

---

- Metrics are not unique, in general there are several homomorphisms
- Admissible transformation  $\Phi$ 
  - ◆  $\Phi \circ m$  is an homomorphism
  - ◆ Mapping between two measures, e.g. length
    - Admissible transformation:  $M' = a * M$
    - Inadmissible transformation:  $M' = a * M + b$

---

## MEASUREMENT SCALES

# Issues

---

- Representation problem
  - ♦ How do we know if a particular empirical relation system has a representation in a given numerical relation system?
- Uniqueness problem
  - ♦ How do we deal with several possible alternative representations (scales) in the same numerical relation system?
- Pragmatic problem
  - ♦ Which is the preferred numerical relation system for a given empirical relation system?

# Relation System richness

---

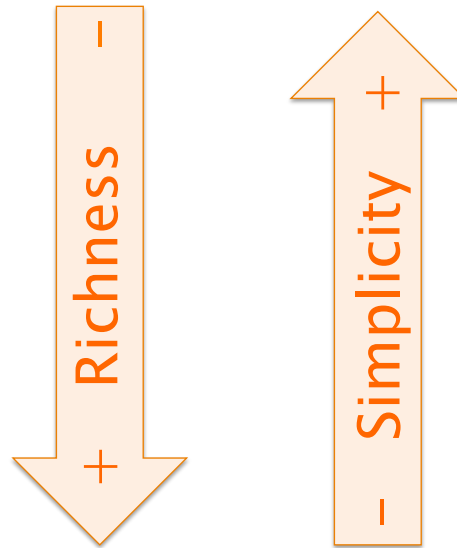
- $RS_A$  is richer than  $RS_B$  if all relations in  $RS_B$  are contained in  $RS_A$
- The richer the empirical system the more sophisticated the scale
- Complex and well understood phenomena require more sophisticated measurement scales



# Scale types

---

- Nominal
- Ordinal
- Interval
- Ratio
- Absolute



# Admissible measure

---

- Measure that is able to represent all the empirical relations
  - ◆ There may exist several admissible measure
    - E.g. Length: inch, cm, feet, meter, miles
- Admissible transformation
  - ◆ Mapping between two admissible measures
  - ◆ The more sophisticated the scale the more restricted the class of admissible transformation
    - E.g. Admissible Length transformation:  $M' = a * M$
    - Inadmissible transformation:  $M' = a * M + b$


# Nominal scale

---

- Places elements in classification schema
- Empirical scale: different classes
  - ♦ No ordering relation
- Any representation based on a set of distinct numbers or symbols is acceptable
  - ♦ No notion of magnitude

# Nominal scale example

---

- Empirical system
  - ♦ Entity: fault
  - ♦ Attribute: artifact type
    - Specification, design, code
- Admissible mapping
  - ♦  $M(x) =$ 
    -  S if x is a specification fault
    - D if x is a design fault
    - C if x is a code fault

# Nominal Statistics

---

- Only a base operation: count
- Available statistics
  - ♦ Frequency (per category)
  - ♦ Mode

# Ordinal scale

---

- Empirical system: classes ordered wrt the attribute
- Acceptable mapping: any mapping preserving the order
  - ♦ Measure represent ranking only
  - ♦ Acceptable transformations are the set of all monotonic mappings
  - ♦  $\langle C_1, C_2, \dots, C_n \rangle \rightarrow \langle a_1, a_2, \dots, a_n \rangle$
  - ♦ Where  $\forall i > j, a_i > a_j$

# Ordinal scale example

---

- Empirical system
  - ♦ Entity: statement
  - ♦ Attribute: agreement
    - Completely disagree, Mostly disagree, Mostly agree, Completely agree
- Admissible mapping
  - ♦  $M(x) =$ 
    - 2 if x is Completely disagree
    - 1 if x is Mostly disagree
    - 1 if x is Mostly agree
    - 10 if x is Completely agree

# Ordinal scale example

---

- Empirical system
  - ♦ Entity: code
  - ♦ Attribute: size class
    - Small, medium, large
- Admissible mapping
  - ♦  $M(x) =$ 
    - 1 if x is small
    - 2 if x is medium
    - 3 if x is large

# Ordinal Statistics

---

- Operations:
  - ♦ Counting
  - ♦ Sorting
- Available statistics
  - ♦ Frequency (per category)
  - ♦ Mode
  - ♦ Rank
  - ♦ Quantiles (Median)

# Interval scale

---

- Empirical system: order and differences between classes
- Acceptable mappings: preserve order and difference
  - ♦ Addition and subtraction make sense
  - ♦ The ratio makes no sense
- Acceptable transformations are affine transformations
  - ♦  $M' = a * M + b$

# Interval scale example

---

- Empirical system
  - ♦ Entity: activity
  - ♦ Attribute: calendar start time
    - Gregorian calendar
    - Months since project begin
- Admissible transformation
  - ♦ PM counts month since project start
    - Jan 1, 2010
  - ♦ CEO uses calendar year
  
  - ♦  $M_{PM} = 12 * (M_{CEO} - 2010)$

# Interval Statistics

---

- Operations:
  - ♦ Counting, sorting
  - ♦ Sum, Difference, Scalar division
- Available statistics
  - ♦ Frequency, Mode, Rank, Quantiles
  - ♦ Mean (Arithmetic Average)
  - ♦ Variance (and derivatives)

# Ratio scale

---

- Preserves ordering, size of intervals, and ratios between entities
- There is a **zero** element
  - ◆ Represents total lack of attribute
  - ◆ Measurement starts at zero and increases at equal intervals: called **units**
  - ◆ All arithmetic can be applied meaningfully to classes in the range of the mapping
- Admissible transformation
  - ◆ Ratio transformation
  - ◆  $M' = a * M$

# Ratio scale example

---

- Empirical system
  - ◆ Entity: person
  - ◆ Attribute: age
    - Years, Months
- Admissible transformation
  - ◆  $M_{\text{Months}} = a * M_{\text{Year}}$ 
    - Where  $a = 12$

# Ratio scale example

---

- Empirical system
  - ♦ Entity: code
  - ♦ Attribute: length
    - LOC
- Admissible transformation
  - ♦  $M_{LOC}$  = lines of code
  - ♦  $M_{Char}$  = characters of code
  - ♦  $M_{Char} = a * M_{LOC}$ 
    - Where a = average chars per line of code

# Ratio Statistics

---

- Operations:
  - ♦ Counting, sorting
  - ♦ Sum, Difference, Scalar division
  - ♦ Division, (Multiplication)
- Available statistics
  - ♦ Frequency, Mode, Rank, Quantiles, Mean (Arithmetic Average), Variance (and derivatives)
  - ♦ Standardized mean, etc.
  - ♦ Geometric mean, etc.



# Absolute scale

---

- Measurement made simply counting items in the entity set
  - ◆ Number of occurrences
  - ◆ Only one possible mapping
  - ◆ All arithmetic analysis is meaningful

# Absolute scale (counter)examples

---

- Empirical system
  - ◆ Entity: project
  - ◆ Attribute: full time staff
    - Number of full time developers
- The attribute definition implies the items to be counted!
  - ◆ Length is not measurable on an absolute scale, # of lines it is
  - ◆ Age is not measurable on absolute scale

# Scales

Scale	Admissible Transformations	Example
Nominal	1-to-1 mapping	Labeling, classifying entities
Ordinal	Monotonic increasing function	Preference, hardness
Interval	$M' = a*M + b$ With: $a > 0$	Relative time, temperature
Ratio	$M' = a*M$ With: $a > 0$	Time interval, length
Absolute	$M' = M$	Counting entities

51

## Meaningful statements

- A statement involving measurement is meaningful if its truth is invariant of transformation of allowable scales

# Meaningful statements

---

- Statements
  - ✓ The number of errors discovered during the integration testing was at least 100
  - ◆ The cost of fixing each error is at least 100 ?
  - ✓ A semantic error takes twice as long to fix as a syntactic error
  - ◆ A semantic error is twice as complex as a syntactic error

# Meaningful statements?

---

- Fred is twice as tall as Jane
- The temperature in Tokyo today is twice that in London
- The difference in temperature between Tokyo and London today is twice what it was yesterday

# Statistical operations

---

- Central tendency

Type	Mean	Median	Mode
Nominal	✗	✗	✓
Ordinal	✗	✓	✓
Interval	✓	✓	✓
Ratio	✓	✓	✓
Absolute	✓	✓	✓

# Objective vs. Subjective

---

- Objective measures do not depend on the environment or the person collecting the measure
  - ♦ A small portion of subjectivity cannot be avoided
- Subjective measures depend on the context where they are collected
  - ♦ Can change according to the person
  - ♦ They reflect the perception and judgment of the person performing the measurement

# Interpretation

---

- If only measure values are available you know nothing
- Interpretation requires a reference to
  - ◆ Target
  - ◆ Benchmark
  - ◆ Time series
  - ◆ Population norm

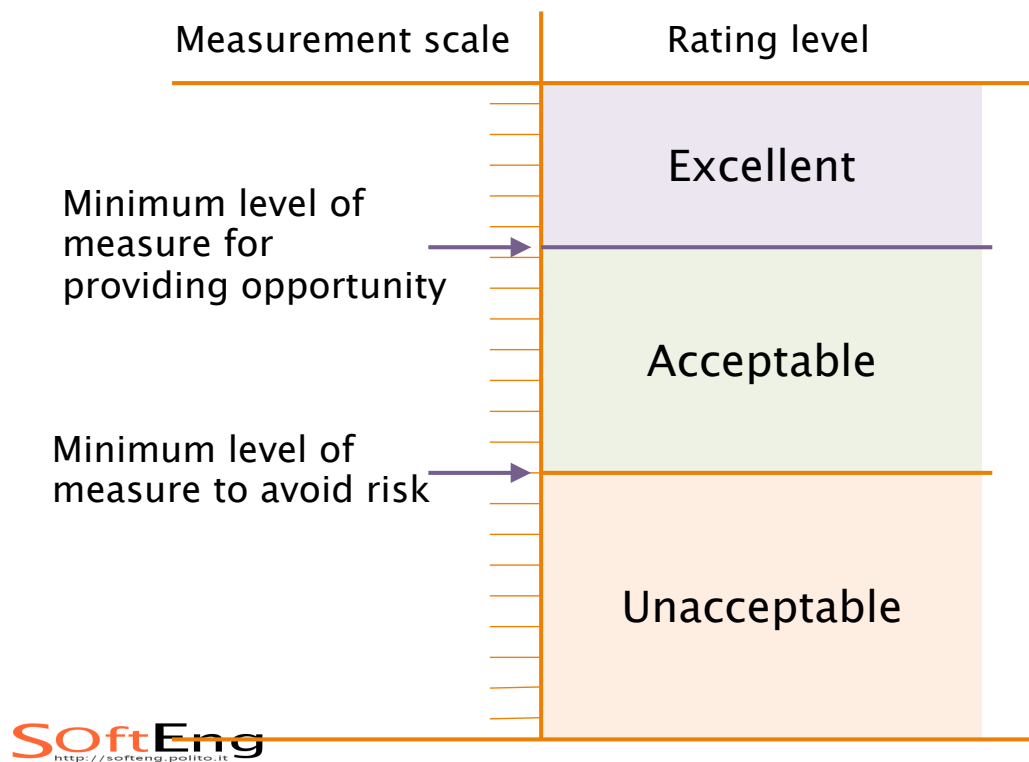
# Interpretation

---

- **Conformance**: compare to a specific business or usage requirement
- **Benchmark**: compare with a benchmark for similar product or system
- **Time series**: observe trend in time
- **Population norms**: compute quantile
  - ◆ Require a db of previous values

# Interpretation: rating

---



59

---

## SOFTWARE MEASURES

60

# Process measures


---

- Duration
  - ◆ Of process or one of its activities
- Effort
  - ◆ Of process or one of its activities
- Number of events
  - ◆ Of a given type
  - ◆ Arising during process or one of its activities
- Subjective measures

# Product measures

---

- External attributes
  - ◆ Reliability
  - ◆ Understandability
  - ◆ Usability
  - ◆ Integrity
  - ◆ Efficiency
  - ◆ Testability
  - ◆ Reusability
  - ◆ Portability



ISO 9126

# Quality myth

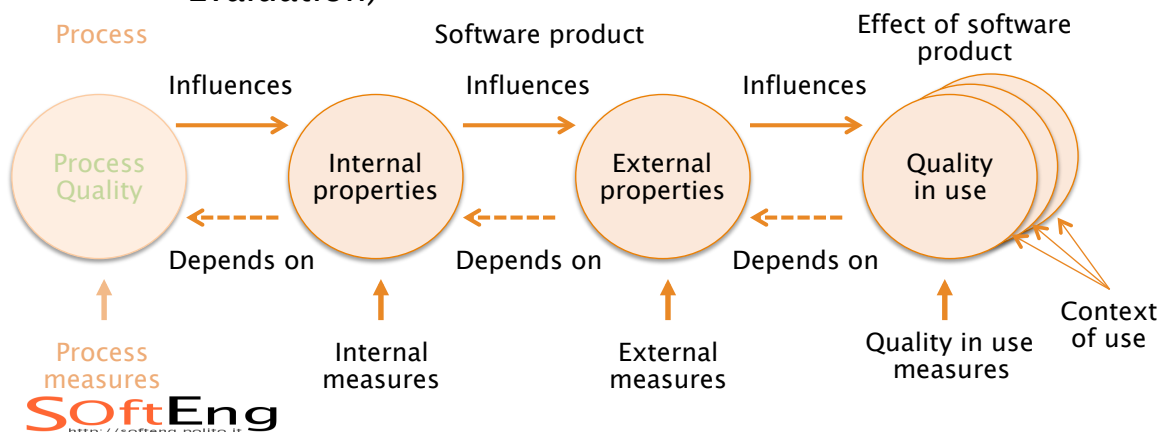
---

- Term used to describe an internal attribute
- Inherently multidimensional
  - ◆ There are several aspects to quality
  - ◆ A single aggregate (indirect) measure of quality implies weighting all different aspects

# ISO 9126

---

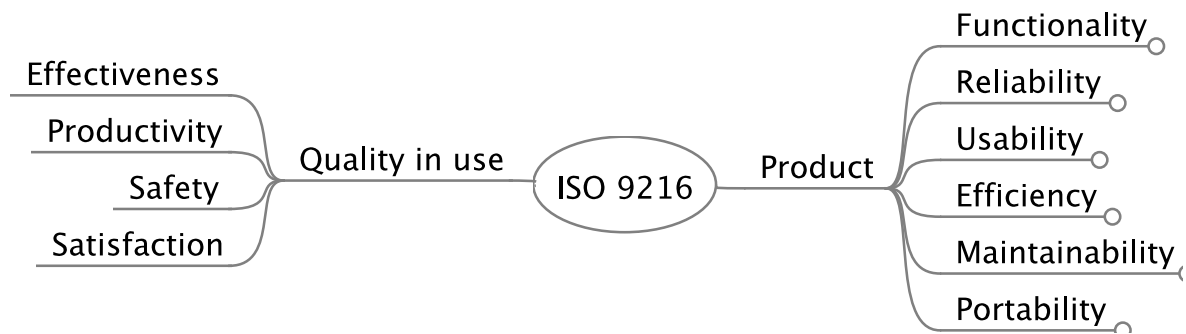
- Software product quality
  - ◆ Issued 1991, revised 2001
  - ◆ Being superseded by ISO/IEC 250xx
    - SQuaRE (Software product Quality Requirements and Evaluation)





# ISO 9126

---



**SoftEng**  
http://softeng.polito.it

---

## ISO 9126 – External metric

---

### ▪ Breakdown avoidance

<b>Purpose</b>	How often can user avoid breakdown of system, even if critical failures occurred?
<b>Method of application</b>	Count the number of breakdowns occurrence with respect to number of failures. If it is under operation, analyze log of user operation history.
<b>Definition</b>	Breakdown avoidance ratio $X = 1 - (A / B)$ A= Number of breakdowns B= Number of failures NOTE: 1.The breakdown means executing of any user task is suspended until system is restarted up, or its control is lost until system is enforced to be shut down. 2. When none or a few failures observed, time between breakdown may be more suitable.
<b>Interpretation</b>	$0 \leq X \leq 1$ The closer to 1.0 is the better.

---

# ISO 9126 – Internal metric

---

- Test coverage

Purpose	How much of the required test cases are covered by the test plan?
Method of application	Count the number of test cases planned and compare it to the number of test cases required to obtain adequate test coverage.
Definition	$X=A/B$  A=Number of test cases designed in test plan and confirmed in review B= Number of test cases required
Interpretation	$0 \leq X$ Where X is the greater the better adequacy

## Product metrics

---

- Internal attributes
  - ◆ Few simple and easy to measure
    - E.g. size
  - ◆ Other controversial
    - E.g. complexity
  - ◆ Automated measurement

# Internal Product Attributes

---

- Methodologies address structuring and improvement of software products in terms of
  - ◆ Development process
  - ◆ Products
    - Typically characterized by internal attributes
- Quality assurance
  - ◆ Internal attributes can be measured during development to predict and control external ones

# Resources

---

- Input for sw development
  - ◆ Personnel
    - Individuals and teams
  - ◆ Materials
    - E.g. office supplies
  - ◆ Tools
    - Both HW and SW
  - ◆ Methods

# Resource metrics

- Magnitude
  - ♦ E.g. How many staff?
- Cost
  - ♦ E.g. Payments for testing tools
- Quality
  - ♦ E.g. Experience of developers

- Productivity =  $\frac{\text{Amount of output}}{\text{Effort input}}$ 
  - ♦ Indirect measure

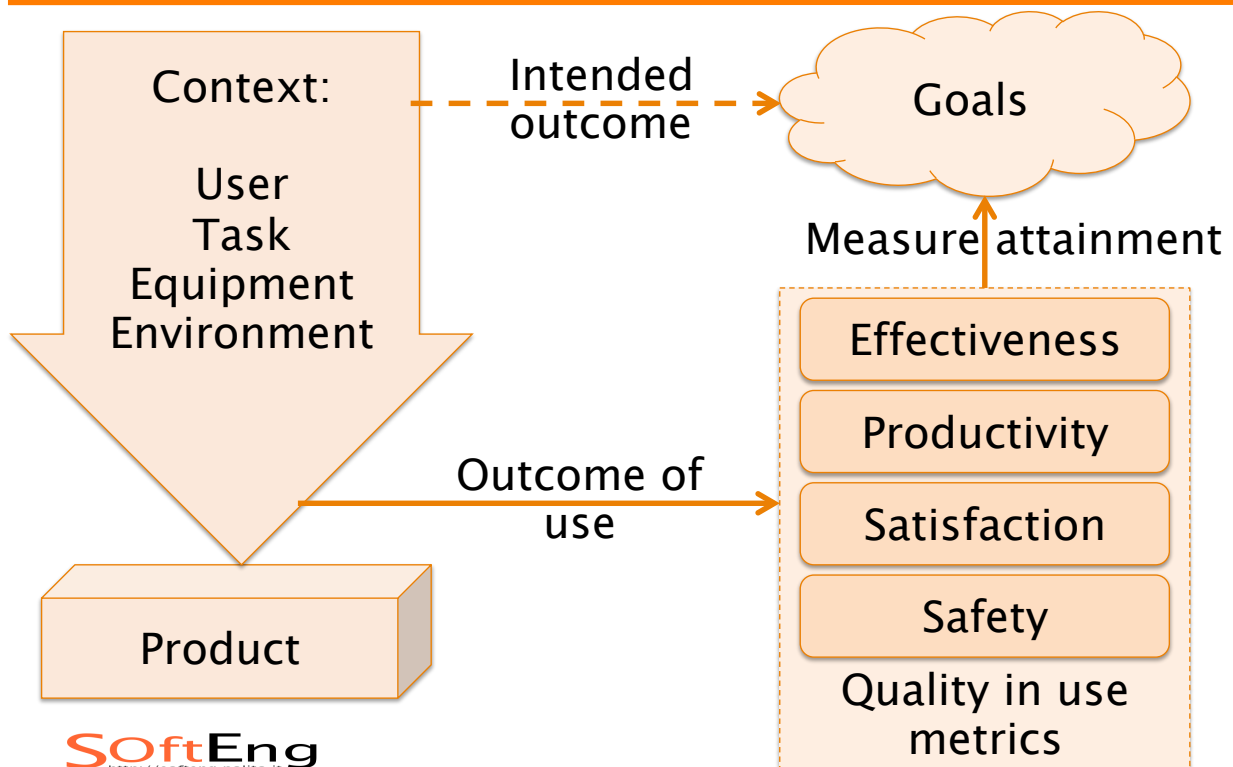
Product

Process

SoftEng  
<http://softeng.polito.it>

71

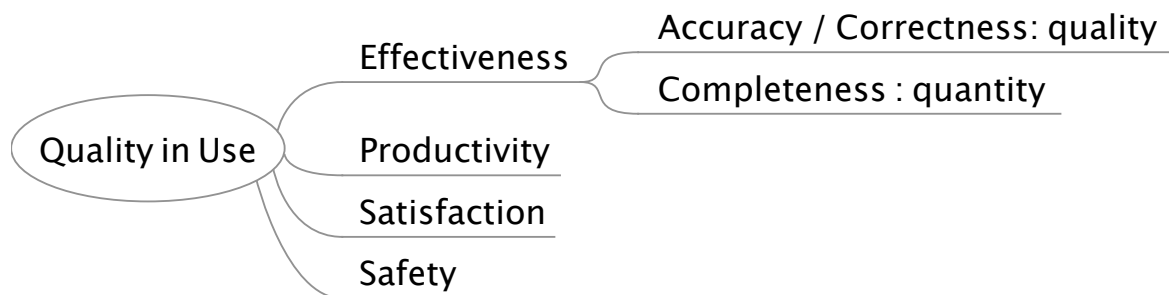
# Quality in use



SoftEng  
<http://softeng.polito.it>

72

# Quality in use



## ISO 9126 – Q in Use metric

- Task effectiveness

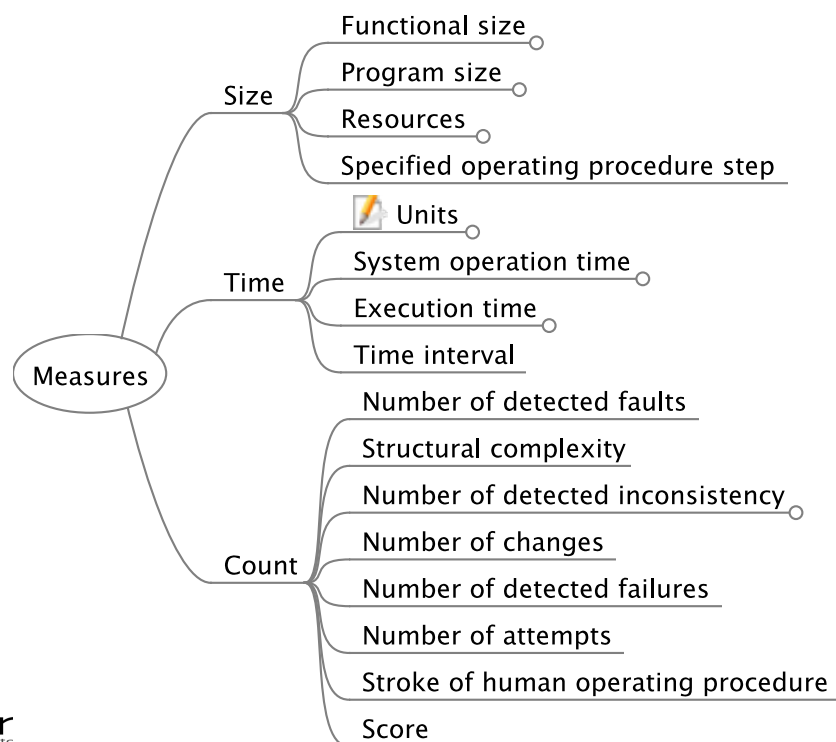
Purpose	What proportion of the goals of the task is achieved correctly?
Method of application	User test
Definition	$M1 =  1 - \sum A_i $ <p><math>A_i</math> = Proportional value of each missing or incorrect component in the task output</p> <p>NOTE: Each potential missing or incomplete component is given a weight <math>A_i</math> based on the extent to which it detracts from the value of the output to the business or user. The scoring scheme is refined iteratively by applying it to a series of task outputs and adjusting the weights until the measures obtained are repeatable, reproducible and meaningful.</p>
Interpretation	$0 \leq M1 \leq 1$ The closer to 1.0 is the better.

---

# COMMON METRICS

## Common measures

---



# Function points

---

- Function Point Analysis, developed by Allan J. Albrecht in the late 1970s
- Several variations
  - ◆ ISO/IEC 19761 (COSMIC method),
  - ◆ ISO/IEC 20926 (IFPUG method)
  - ◆ ISO/IEC 20968 (Mk II method),
  - ◆ ISO/IEC 24570 (NESMA method), and
  - ◆ ISO/IEC 29881 (FiSMA method).

# COSMIC FP – Principles

---

- Software interacts with its users across a boundary, and with storage
- User requirements can be mapped into unique functional processes.
- Each functional process consists of sub-processes: a data movement or a data manipulation.
- A data movement moves a single data group .
  - ◆ Entry: data from user to system.
  - ◆ Exit data from system to user.
  - ◆ Write data from system to persistent storage.
  - ◆ Read data from persistent storage to system.
- Data group: set of attributes that describe a single object of interest
- Each process is started by its triggering Entry data movement.

# Lines Of Code (LOC)

---

- Most intuitive
  - ♦ Count the number of lines of code
- Operational aspects
  - ♦ What to include/exclude in the count?
  - ♦ How to deal with complex lines?

## LOC – Operational aspects

---

- Inclusion/exclusion
  - ♦ Executable lines
  - ♦ Declarations
  - ♦ Comments
  - ♦ COTS
  - ♦ Automatically generated code
  - ♦ Reused code
- Multiple instructions on a line
  - ♦ Number of statements
  - ♦ Number of lines



# LOC: Pros & Cons

---

- Easy to understand 😊
- Hard to measure precisely 😞
  - ◆ Easy of an approximate measure 😊
    - e.g. `wc -l *.c`
- Very widely used 😊
  - ◆ Several predictive models use LOC 😊
- If measures productivity it does not favors well structured programs 😞

# Mc Cabe Cyclomatic Complexity

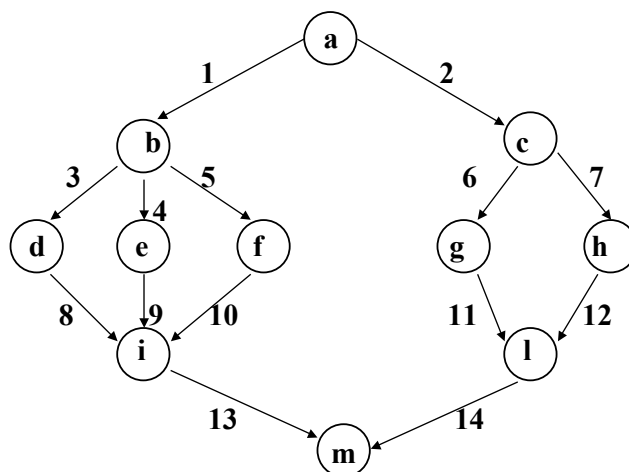
---

- Complexity of the control flow
- Control flow is represented as a Control Flow Graph (CFG)
- $V(G)$  is the number of base paths in  $G$ 
  - ◆ The number of linearly independent paths from initial node to final node

# Cyclomatic number

- If  $G$  is a strongly connected graphs
  - ♦  $V(G) = \#E - \#N + 1$
- A typical CFG is not strongly connected, unless we add an edge from the final to the initial node
  - ♦  $V(G) = \#E - \#N + 2$

# Cyclomatic complexity



$$V(G) = E - N + 2 = 14 - 11 + 2 = 5$$

C1: 1,3,8,13
C2: 1,4,9,13
C3: 1,5,10,13
C4: 2,6,11,14
C5: 2,7,12,14

# McCabe Pros & Cons

---

- Well defined from a mathematical point of view
- Typically strongly correlated with LOC
- Focus on code complexity
  - ◆ Disregards data-related complexity

# Design Metrics – CK

---

- Chidamber and Kemerer [TSE94]:
- Weighted Methods per Class (WMC)
  - ◆ count of methods in each class
- Number Of Children per class (NOC)
  - ◆ number of immediate sub-classes of a class
- Depth of Inheritance Tree (DIT):
  - ◆ maximum inheritance path from the class to the root class
- Coupling Between Object classes (CBO)
  - ◆ number of classes to which a class is coupled

# Design Metrics – CK

---

- Response For a Class (RFC)
  - ♦ Sum of cardinalities of
    - methods in the class
    - remote methods directly called by methods of the class
- Lack of Cohesion in Methods (LCOM)
  - ♦  $LCOM = P - Q$ , if  $P > Q$   
= 0 otherwise
  - ♦ Where
    - $Q$  = # pairs of methods sharing attributes
    - $P$  = # pairs of methods not sharing attributes

# LCOM – Henderson–Sellers

---

- Alternative definition of LCOM

$$LCOM2 = 1 - \frac{\sum mA_i}{m \cdot a}$$

- Where
  - ♦  $m$ : number of methods in class
  - ♦  $a$ : number of attributes in class
  - ♦  $mA_i$ : num. of methods using attribute  $A_i$

# CK – Pros & Cons

---

- Theoretical validation lacking ☹️
- Empirical validation lacking ☹️
- Not all metrics can be easily computed
  - ◆ RFC e LCOM need implementation details
    - Design or code metrics?

---

## DATA QUALITY

# Data Quality

---

- Decisions taken on the basis of indicators are as good as the quality of the indicators themselves
- Data quality is a key factor

# ISO – SQuaRE

---

2503x Quality Requirements	2501x Quality Model	2504x Quality Evaluation
	2500x Quality Management	
	2502x Quality Measurement	



# Bibliography

---

- Stevens, S. S. (June 7, 1946). "On the Theory of Scales of Measurement". *Science* 103 (2684): 677–680
- Roberts, F. (1979). "Measurement Theory with Applications to Decision Making, Utility, and the Social Sciences", Addison–Wesley.
- N. Fenton (1994). "Software Measurement: A Necessary Scientific Basis." In *IEEE Transactions on Software Engineering*, 20(3):199–206, March 1994.
- N. Fenton, S. Pfleeger. "Software Metrics: A Rigorous Approach". PWS Publishing, 1998.
- ISO (2007). Systems and software engineering — Measurement process, ISO/IEC 15939:2007(E)

# Bibliography

---

- S.R. Chidamber, C.F. Kemerer. "A Metrics Suite for Object–Oriented Design" *IEEE Transactions on Software Engineering* 20(6), 1994.
- T. McCabe "A complexity measure" *IEEE Transactions on Software Engineering* 2(4), 1976.
- V. Basili, L. Briand, W. Melo. "A Validation of Object–Oriented Design Metrics as Quality Indicators" *IEEE Transactions on Software Engineering* 22(10), 1996