SENG 421: Software Metrics
Measurement Theory
(Chapter 2)

Department of Electrical & Computer Engineering, University of Calgary

B.H. Far (far@ucalgary.ca)
http://www.enel.ucalgary.ca/People/far/Lectures/SENG421/02/
Contents

- Metrology
- Property-oriented measurement
- Meaningfulness in measurement
- Scale
- Measurement validation
- Object-oriented measurement
- Subject-domain-oriented measurement
Metrology

- Metrology is the science of measurement.
- Metrology is the basis for empirical science and engineering in order to bring knowledge under general laws, i.e., to distil observations into formal theories and express them mathematically.
- Measurement is used for formal (logical or mathematical, orderly and reliable) representation of observation.
Two Problem Categories

- Components of a measurement system:
  \[ m = \langle \text{attribute}, \text{scale}, \text{unit} \rangle \]
  - Attribute is what is being measured (e.g., size of a program)
  - Scale is the standard and scope of measurement (e.g., nominal, ordinal, ratio scale, etc.)
  - Unit is the physical meaning of scale (e.g., a positive integer, a symbol, etc.)

- Determining the value of an attribute of an entity.
- Determining the class of entities to which the measurement relates.
Empirical Relations

- Empirical relation preserved under measurement $M$ as numerical relation

Figure from Fenton’s Book
Real, Empirical & Formal Worlds

Real World <-> Empirical World

Measurement

Modeling & Verification

Mapping

Mathematical (logical) Model

Scales & Units
Real, Empirical & Formal Worlds

Entity A

Entity B

Measurement
height

M(A) > M(B)

Mapping

Scale: ratio
Unit: cm

Modeling & Verification
Measurement: Activities /1

- **Problem definition**
  - Defining measurement problem
  - Designating set of entities forming the target of measurement
  - Identifying key attributes for the entities

- **Identifying scales**
  - Identifying scales for which the attributes can be measured

- **Forming the empirical relational system**
  - Mapping entities and their selected attributes to numbers or values on the scales
Measurement: Activities /2

- **Modeling**
  - Developing mathematical (logical) representation of the entities and their attributes

- **Defining the formal relational system**
  - Mapping the empirical relational system to the formal model

- **Verifying the results of measurement**
  - Verifying whether the measurement results reflect the properties of the entities, attributes and relationships
Empirical Relational System

- $E = \{A, R, O\}$
- $A = \{a, b, c, \ldots, z\}$ set of real world entities
- $k$: property for each element of $A$
- $A = \{a, b, c, \ldots, z\}$ is the model of $A$ which describes each element of $A$ in terms of the property $k$
- Empirical observations defined on the model set $A$
  - Set of n-ary relations: $R$, e.g., $X$ is taller than $Y$
  - Set of binary operations: $O$, e.g., $X$ is tall
Formal Relational System

\[ F = \{ A', R', O' \} \]

- \( F \) should satisfy the following conditions:
  - Capable of expressing all relations and operations in \( E \).
  - Support the meaningful conclusions from the data.
  - Mapping from \( E \) to \( F \) must represent all the observations, preserving all the relations and operations of \( E \).
Example 1

Ranking 4 software products based on user preferences

- **Problem definition:** ranking the products (entities) A, B, C and D based on user preferences
- **Scale:** A single 0-100% linear scale
- **Empirical relational system:** represented by the table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>80</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>-</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>90</td>
<td>95</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>50</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

\((A, B) = 80\) means %80 of the users preferred product A to B

*Example from Fenton’s Book*
Example 1 /2

- **Modeling:**
  - Valid pairs are those having the value greater than 60.
  - If for a pair \((A,B)\), more than 60% of users prefer \(A\) to \(B\) then \(A\) is “definitely better” than \(B\).
    \[
    \text{If } M(x,y) > 60\% \text{ then } p(x) > p(y)
    \]

- **Formal relation system:**
  - If \(p(x) > p(y)\) and \(p(y) > p(z)\)
  - Then \(p(x) > p(z)\)

- **Verification:**
  - Valid pairs \((C,A), (C,B), (C,D), (A,B)\) and \((A,D)\)
  - No conflict between the data collected and the model
Example 1 /3

- Verification fails if the data was collected as shown here because

\[ M(B,C) > 60\% \text{ then } p(B) > p(C) \]
\[ M(C,A) > 60\% \text{ then } p(C) > p(A) \]
\[ M(A,B) > 60\% \text{ then } p(A) > p(B) \]

- There is a conflict between the real and formal model and the model must be revised.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>80</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>-</td>
<td>95</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>90</td>
<td>5</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>50</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>
Example 2

- Entity: software failure
- Attribute: criticality
- Three types of failure is observed:
  - Delayed response
  - Incorrect output
  - Data loss
- There are 3 failure classes in E
  - Delayed response (R1)
  - Incorrect output (R2)
  - Data loss (R3)
Example 2 / 2

- Measurement mapping

Figure from Fenton’s Book
Next we would like to add a new binary relation:

\[ x \text{ is more critical than } y \]

Each data loss failure (x in R3) is more critical than incorrect output failure (y in R2) and delayed response failure (y in R1).

Each incorrect output failure (x in R2) is more critical than delayed response failure (y in R1).

The model should be revised to account for this new binary relation.
Example 2 /4

- Revised measurement mapping

- $R_1$
  (each point in here corresponds to a delayed-response failure)

- $R_2$
  (each point in here corresponds to an incorrect-output failure)

- $R_3$
  (each point in here corresponds to a data-loss failure)

Figure from Fenton’s Book
**Completeness**

- Components of a measurement system: \( m = \langle \text{attribute, scale, unit} \rangle \)
- Attribute is what is being measured (e.g., size of a program)
- Scale is the standard and scope of measurement (e.g., ratio)
- Unit is the physical meaning of scale (e.g., a positive integer)

**Uniqueness**

- A measurement result should be unique and match with the scale and units
Measurement: Properties /2

- **Extendibility**
  - Two or more formal measures mapping to the same entity are compatible using explicit compatibility relation (e.g., cm and inch used to measure length and 1 in = 2.54 cm)

- **Discrete Differentiability**
  - The minimum unit of measurement scale is used to determine the differential rate of the measurement system.
Measurement: Properties /3

- **Deterministic or Probabilistic**
  - Measurement system should be either deterministic (i.e., lead to same results under same conditions) or probabilistic (e.g., productivity per hour, reliability)

- **Quantitative or Qualitative**
  - Result of measurement is either quantitative (represented by number values) or qualitative (represented by qualitative values or range intervals)
Measurement: Direct & Indirect

- Direct measurement of an attribute of an entity involves no other attribute or entity. E.g., software length in terms of lines of code.

- Indirect measurement is useful in making visible the interaction between direct measurements. E.g., productivity, software quality.

- Indirect measurement is particularly helpful when the size of empirical measures is quite big (i.e., many relations and many entities) or the cost of direct measurement is high.
Example: Direct & Indirect

- In a software system, measuring number of faults (i.e., direct measurement) leads to identification of 5 problem areas.

- However, measuring faults per KLOC (i.e., indirect measurement) leads to identification of only one problem area.
Measurement: Size of Empirical Set

- **Example:** measuring popularity of 4 software products

  \[ E = \{A, R, O\} \]
  - \(|A| = 4\)
  - \(|R| = 3\)
    - \(r_1: x \text{ is more popular than } y\)
    - \(r_2: x \text{ is less popular than } y\)
    - \(r_3: \text{ indifferent}\)

- The total number of possible measures is:
  \[ 4 \times (4 - 1) \times 3 = 36 \]

- The size will grow when the number of elements and relations grow.
Software Measurement: Scale
Scale: Informal Definition

- A **measurement scale** is a set of predefined symbols or values in order to represent certain common measures.

- A **scale** is an abstract measurement tool for measuring defined common attributes of entities.
Scale: Formal Definition

- Assume that the followings are given:
  - \( A = \{a, b, c, \ldots, z\} \) set of real world entities
  - \( k \): property for each element of \( A \)
  - \( A = \{a, b, c, \ldots, z\} \) is the model of \( A \) which describes each element of \( A \) in terms of the property \( k \)
  - The empirical and formal relational systems \( E \) and \( F \)
  - \( m \), which is a mapping \( A \rightarrow A' \)

- Then \( S = (E, F, m) \) is called the scale of measurement for the property \( k \) of the target set \( A \)

- If \( F \) is defined over a subset of real numbers, within the scale, the measurement maps the key property of each object of the target set into a number.
Scales (History)

- In the early 1940’s, the Harvard psychologist Stanley Smith Stevens coined the terms *nominal*, *ordinal*, *interval*, and *ratio* to describe a hierarchy of measurement scales and classify statistical procedures according to the scales they were adopted.

- Stevens’ scales, were first presented in his 1946 article “On the theory of scales of measurement” [Stevens, 1946]. They have been adopted by most of the statistics textbooks and have consequently been influencing statistical experiments to date.
Scales (Summary)

- Nominal
- Ordinal
- Interval
- Ratio

- Level 0: Categorized data
- Level I: Ordered categories of data
- Level II: Measured intervals
- Level III: True zero
Measurement Scale: Questions

- How do we know if a scale appropriately represents the relationships between measured properties of entities? (representation problem)
  → Has to do with the validity of the measure

- What do we do when we have several different scales for the same measure? (uniqueness problem)
  → Has to do with transformation between scales (→ unit!)

- For a defined scale, what arithmetic operations make sense for measurement values?
  → Has to do with meaningfulness of measurement-based statements
Scale Types

- Objective (regular) scales
  - Nominal, ordinal, interval, ratio, absolute

- Subjective scales
  - Likert-Type scale (Evaluation-Type, Frequency-Type, Agreement-Type)
  - Semantic differential scale
  - Summative scale
Regular Scales

- The scale \((E,F,m)\) is regular if and only if
  - for every other scale \((E,F,g)\), and
  - for all \(a\) and \(b\) in \(A\),
    \[ m(a) = m(b) \text{ implies } g(a) = g(b) \]

- This is a rephrase for uniqueness property along various scales.

- If two objects measured on one scale yield to the same measure for the property \(k\), the same holds on any other scale.
Nominal Scales

- Define classes or categories, and then place each entity in a particular class or category, based on the value of the attribute.

- **Properties:**
  - The empirical relation system consists only of different classes; there is no notion of ordering among the classes.
  - Any distinct numbering or symbolic representation of the classes is an acceptable measure, but there is no notion of magnitude associated with the numbers or symbols.
  - Nominal-scale measurement places elements in a classification scheme. The classes are not ordered; even if the classes are numbered from 1 to n for identification, there is no implied ordering of the classes.
Example: Nominal Scale

- Classification of cars based on their colour
- Classification of faults in a software:
  - Specification fault
  - Design fault
  - Coding fault

Entity

Attr

Car

Colour

C-C1 1 White
C-C2 2 Yellow
C-C3 3 Red
C-C4 4 Blue
C-C5 5 Green
C-C6 6 other

Measure (Car Colour) ∈ {“1”, “2”, “3”, “4”, “5”, “6”}
{White, ..., other}
Objective Scale: Types /2

Ordinal Scales

- The ordinal scale is useful to augment the nominal scale with information about an ordering of the classes or categories.

- **Properties:**
  - The empirical relation system consists of classes that are ordered with respect to the attribute.
  - Any mapping that *preserves the ordering* (that is, any monotonic function) is acceptable.
  - The numbers represent ranking only, so addition, subtraction, and other arithmetic operations have no meaning.
Example: Ordinal Scale

- $M(x) > M(y)$ then $M'(x) > M'(y)$
- Measuring complexity of many software modules by defining 5 complexity classes
  - Trivial
  - Simple
  - Moderate
  - Complex
  - Incomprehensible

Measure (Defect Severity) $\in \{S1, \ldots, S4\}$
{1, ..., 4}
Objective Scale: Types /3

Interval Scales

- Interval scale carries more information than ordinal and nominal scale. It captures information about the size of the intervals that separate the classes, so that we can in some sense understand the size of the jump from one class to another.

- **Properties:**
  - An interval scale *preserves order*, as with an ordinal scale.
  - An interval scale *preserves differences but not ratios*. That is, we know the difference between any two of the ordered classes in the range of the mapping, but computing the ratio of two classes in the range does not make sense.
  - Addition and subtraction are acceptable on the interval scale, but not multiplication and division.
Example: Interval Scale

- **M’ = aM + b**
- Temperature ranges in Celsius and Fahrenheit
- Project scheduling
  - Requirement analysis: 3 weeks
  - Design: 4 weeks
  - Coding: 4 weeks
  - Testing starts after coding is done
  - When testing starts? After 11 weeks

Measure (Engine Temperature) \( \in [\text{min}, \text{max}] \)

<table>
<thead>
<tr>
<th>Entity</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine</td>
<td></td>
</tr>
<tr>
<td>Attr</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-T1</td>
<td>-20</td>
</tr>
<tr>
<td>E-T2</td>
<td>-10</td>
</tr>
<tr>
<td>E-T3</td>
<td>0</td>
</tr>
<tr>
<td>E-T4</td>
<td>10</td>
</tr>
<tr>
<td>E-T5</td>
<td>20</td>
</tr>
</tbody>
</table>

- Measure (Engine Temperature) \( \in [\text{min}, \text{max}] \)
Objective Scale: Types

Ratio Scales

- Sometimes we would like to be able to say that one liquid is twice as hot as another, or that one project took twice as long as another. This needs the ratio scale, which is the most useful scale of measurement, and quite common in the physical sciences.

- **Properties:**
  - It is a measurement mapping that **preserves ordering, preserves size of intervals** between entities, and **preserves ratios** between entities.
  - There is a zero element, representing total lack of the attribute.
  - The measurement mapping must start at zero and increase at equal intervals, known as units.
  - All arithmetic can be meaningfully applied to the classes in the range of the mapping.
Example: Ratio Scale

- Measuring length, distance, etc. preserves $M' = aM$
- Measuring execution time of a program

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attr</th>
<th>Prog</th>
<th>Exec Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-E1</td>
<td>0</td>
<td>0.001</td>
<td>1</td>
</tr>
<tr>
<td>P-E2</td>
<td>0.001</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>P-E3</td>
<td>0.002</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P-E4</td>
<td>0.003</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>P-E5</td>
<td>0.004</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Measure (Progr. Exec. Time) $\in [0, \infty)$
Absolute Scales

The absolute scale is the most restrictive of all. For any two measures, M and M', there is only one admissible transformation: the identity transformation.

Properties:

- The measurement for an absolute scale is made simply by counting the number of elements in the entity set.
- The attribute always takes the form “number of occurrences of x in the entity.”
- There is only one possible measurement mapping.
- All arithmetic analysis of the resulting count is meaningful.
Example: Absolute Scale

- $M' = M$
- Number of failures observed in a module is absolute but the reliability is not.
- Number of people working on a project is absolute but their productivity is not.

**Entity**

**Attr**

**Module**

**#Defects**

<table>
<thead>
<tr>
<th>Module</th>
<th>#Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-D1</td>
<td>0</td>
</tr>
<tr>
<td>M-D2</td>
<td>1</td>
</tr>
<tr>
<td>M-D3</td>
<td>2</td>
</tr>
<tr>
<td>M-D4</td>
<td>3</td>
</tr>
<tr>
<td>M-D5</td>
<td>4</td>
</tr>
<tr>
<td>M-D6</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Measure (Module Defect Count) $\in \mathbb{IN}_0$
Scale Types (Summary)

- **Nominal scale:** classification of objects, where the fact that objects are different is preserved.
- **Ordinal scale:** objects are ranked/ordered according to some criteria, but no information about the distance between the values is given.
- **Interval scale:** differences between values are meaningful.
- **Ratio scale:** there is a meaningful “zero” value, ratios between values are meaningful.
- **Absolute scale:** no transformation (other than identity) is meaningful.

Measure (Attribute) is well-defined, if scale and unit are clearly specified; specification of the unit makes the measure unambiguous!
## Measurement Scale Types

<table>
<thead>
<tr>
<th>Scale Type</th>
<th>Characterization</th>
<th>Example (generic)</th>
<th>Example (SE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td>Divides the set of objects into categories, with no particular ordering among them</td>
<td>Labeling, classification</td>
<td>Name of programming language, name of defect type</td>
</tr>
<tr>
<td>Ordinal</td>
<td>Divides the set of entities into categories that are ordered</td>
<td>Preference, ranking, difficulty</td>
<td>Ranking of failures (as measure of failure severity)</td>
</tr>
<tr>
<td>Interval</td>
<td>Comparing the differences between values is meaningful</td>
<td>Calendar time, temperature (Fahrenheit, Celsius)</td>
<td>Beginning and end date of activities (as measures of time distance)</td>
</tr>
<tr>
<td>Ratio</td>
<td>There is a meaningful “zero” value, and ratios between values are meaningful</td>
<td>Length, weight, time intervals, absolute temperature (Kelvin)</td>
<td>Lines of code (as measure of attribute “Program length/size”)</td>
</tr>
<tr>
<td>Absolute</td>
<td>There are no meaningful transformations of values other than identity</td>
<td>Object count</td>
<td>Count (as measure of attribute “Number of lines of code”)</td>
</tr>
</tbody>
</table>
Measurement Scale – Summary

- There are 5 types of measurement scales
- The type of the measurement scale determines
  - how measurement data can be treated
  - whether statements involving measurement data are meaningful
Commonly Used Subjective Measurement Scales

- Likert-Type Scale
  - Evaluation-Type
  - Frequency-Type
  - Agreement-Type
- Semantic Differential Scale
- Summative Scale
Likert Type Scales /1

- **Evaluation-type** [Spe92]

  Example:
  - Familiarity with and comprehension of the software development environment (e.g., compiler, code generator, CASE tools):
    - Little
    - Unsatisfactory
    - Satisfactory
    - Excellent
Likert Type Scales /2

- **Frequency-type** [Spe92]

- Example:
  - Customers provided information to the project team
    - e.g., during interviews, when given questionnaires by the project staff, when presented with a “system walkthrough”, and/or when they are asked to provide feedback on a prototype:
      - Never
      - Rarely
      - Occasionally
      - Most of the time
Likert Type Scales /3

- **Agreement-type** [Spe92]

Example:
- The tasks supported by the software at the customer site were undertaking numerous changes during the project:

  - Strongly Agree
  - Agree
  - Disagree
  - Strongly Disagree
Semantic Differential Scale

- Items which include semantic opposites
- Example:
  - Processing of requests for changes to existing systems: the manner, method, and required time with which the MIS staff responds to user requests for changes in existing computer-based information systems or services.
Assigning Numbers to Scale Responses /1

- Likert-Type Scales:
- Ordinal Scale
- But: Often the distances between the four response categories are approximately (conceptually) equidistant [Spe76][Spe80] and thus are treated like approximate interval scales

- Strongly Agree \(\rightarrow\) 1
- Agree \(\rightarrow\) 2
- Disagree \(\rightarrow\) 3
- Strongly Disagree \(\rightarrow\) 4
Assigning Numbers to Scale Responses /2

- Semantic Differential Scale:
- Ordinal scale, but again, often treated as interval scales
Assigning Numbers to Scale
Responses /3

- **Summative Scale:**
  - A summative scale is constructed by forming an unweighted sum of $n$ items:
    \[
    Y = X_1 + X_2 + \ldots + X_n
    \]
  - This is commonly considered to be an approximately interval scale, but controversial (see [McC81][GL89] for two different views)
  - If the $X$ items are on a Likert-type scale then there is a good empirical basis for treating this as an approximately interval scale
Exercise 1

Determine the best “scale” for each of the following measurements using each of the Nominal, Ordinal, Interval and Ratio scales only once.

- Measuring execution time of a program **Ratio**
- Classification of objects based on their color **Nominal**
- Measuring duration of various phases of projects (Project scheduling) **Interval**
- Measuring complexity of many software modules by defining 4 complexity classes (Trivial, Simple, Moderate, Complex) **Ordinal**
- Measuring complexity of many software modules by defining cyclomatic complexity metrics
Exercise 1 (cont’d)

- Software products categorized according to their compatibility with the operating system (i.e. Windows, Linux, MacOS, DOS, etc.). Nominal
- Internet services categorized according to their relevant technologies (i.e. dial-up, DSL, high-speed, wireless, etc.) Nominal
- Measuring attitudes towards an Internet service (say, on an n-point rating, n = 0 to 10). Ordinal
Exercise 1 (cont’d)

- Measuring attitude towards Internet services, if the evaluation scores are numerically meaningful so the difference between a rate of 3 and a rate of 6 is exactly the same as the difference between a rate of 7 and a rate of 10. Interval

- Measuring attitude towards Internet services, if the differences between the data values are numerically meaningful and equal. In addition, a score of zero implies either the full absence of the service being evaluated or the full dissatisfaction with it. Ratio
Exercise 2

Suppose that you are asked to study various software development tools and recommend the best three to your company. The following table shows a list of available development tools.
## Exercise 2 (cont’d)

<table>
<thead>
<tr>
<th>Tool Name/Vendor</th>
<th>Languages Supported</th>
<th>Platforms</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bean Machine IBM</td>
<td>Java</td>
<td>Windows, OS2, Unix</td>
<td>Best: Visual applet and JavaBean generation</td>
</tr>
<tr>
<td>CodeWarrior Pro Metrowerks</td>
<td>Java, C, C++, Pascal</td>
<td>Unix, Windows, Mac</td>
<td>Best: if you need to support Unix, Windows, and Mac platforms</td>
</tr>
<tr>
<td>Java Workshop Sun Microsystems</td>
<td>Java</td>
<td>Solaris, Windows</td>
<td>Better: Written 100% in Java; tools based on a web browser metaphor</td>
</tr>
<tr>
<td>JBuilder Imprise</td>
<td>Java</td>
<td>Windows, AS400</td>
<td>Better: database support</td>
</tr>
<tr>
<td>Visual Cafe for Java Symantec</td>
<td>Java</td>
<td>Windows</td>
<td>Good: multithreaded debugger</td>
</tr>
<tr>
<td>VisualAge IBM</td>
<td>Java</td>
<td>Unix, Windows</td>
<td>Good: includes incremental compiler and automatic version control</td>
</tr>
<tr>
<td>Visual J++ Microsoft</td>
<td>Java</td>
<td>Windows</td>
<td>Fair: All the bells and whistles for Windows</td>
</tr>
</tbody>
</table>
Exercise 2 (cont’d)

What are the entities, attributes and their values in your model?

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Tool</td>
<td>Language supported</td>
<td>Java, C, C++, Pascal</td>
</tr>
<tr>
<td></td>
<td>Platform</td>
<td>Win, Unix, Mac, OS2, AS400</td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Fair, Good, Better, Best</td>
</tr>
</tbody>
</table>
Exercise 2 (cont’d)

- What is the best scale for each of the attributes you defined?

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attribute</th>
<th>Value</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>Tool</td>
<td>Language supported</td>
<td>Java, C, C++, Pascal</td>
</tr>
<tr>
<td>Platform</td>
<td></td>
<td>Win, Unix, Mac, OS2, AS400</td>
<td>Nominal</td>
</tr>
<tr>
<td>Feature</td>
<td></td>
<td>Fair, Good, Better, Best</td>
<td>Ordinal</td>
</tr>
</tbody>
</table>
Are the following statements meaningful?

1. Peter is twice as tall as Hermann
2. Peter’s temperature is 10% higher than Hermann’s
3. Defect X is more severe than defect Y
4. Defect X is twice as severe as defect Y
5. The cost for correcting defect X is twice as high as the cost for correcting defect Y
6. The average temperature of city A (15 ºC) is twice as high as the average temperature of city B (30 ºC)
7. Project Milestone 3 (end of coding) took ten times longer than Project Milestone 0 (project start)
8. Coding took as long as requirements analysis
Are the following statements meaningful?

1. “Peter is twice as tall as Hermann”
2. “Peter’s temperature is 10% higher than Hermann’s”
3. “Defect X is more severe than defect Y”
4. “Defect X is twice as severe as defect Y”
5. “The cost for correcting defect X is twice as high as the cost for correcting defect Y”
6. The average temperature of city A (30 °C) is twice as high as the average temperature of city B (15 °C)
7. “Project Milestone 3 (end of coding) took ten times longer than Project Milestone 0 (project start)”
8. “Coding took as long as requirements analysis”
Software Measurement: Validation & Accuracy
Measurement vs. Prediction

- **Measurement systems** are used to assess existing entities by numerically characterizing one or more of its attributes.

- **Prediction systems** are used to predict some attributes of a future entity, involving a mathematical model with associated prediction procedures.
  - **Deterministic prediction system**: The same output will be generated for a given input.
  - **Stochastic prediction system**: The output for a given input will vary probabilistically. E.g., weather forecast.
Measurement Validation

- Validation question: Does the measure used capture the information that it was intended for?

- A measure is valid if it accurately characterizes the attribute it claims to measure.

- A prediction system is valid if it makes accurate predictions.
Validating Measures

- **Definition:** The process of ensuring that the measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied.

- **Example:** Measuring program length
  - Any measure of length should satisfy conditions such as:
    
    \[ m(p_1; p_2) = m(p_1) + m(p_2) \]
    
    If the length of \( p_1 \) is greater than \( p_2 \), any measure of length should satisfy
    
    \[ m(p_1) > m(p_2) \]
Validating Prediction Systems

- **Definition:** The process of establishing the accuracy of the prediction system by empirical means, i.e., by comparing model performance with known data in the given environment.

- **Example:** Software Reliability
  - Using tools such as CASRE to fit the failure data into the reliability model.
  - Accuracy of estimation of the failure intensity $\lambda$ depends on the number of failures experienced (i.e., the sample size).
  - Good results in estimating failure intensity are generally experienced for programs with 5,000 or more developed source lines.
Classes of Prediction Systems

- **Class 1:** Using internal attribute measures of early life-cycle products to predict measures of internal attributes of later life-cycle products.
  - **Example:** measures of size, modularity, and reuse of a specification are used to predict size of the final code.

- **Class 2:** Using early life-cycle process attribute measures and resource-attribute measures to predict measures of attributes of later life-cycle processes and resources.
  - **Example:** the number of faults found during formal design review is used to predict the cost of implementation.
Classes of Prediction Systems

- **Class 3:** Using internal product-attribute measures to predict process attributes.
  - **Example:** measures of structuredness are used to predict time to perform some maintenance task, or number of faults found during unit testing.

- **Class 4:** Using process measures to predict later process measures.
  - **Example:** measures of failures during one operational period are used to predict likely failure occurrences in a subsequent operational period. In examples like this, where an external product attribute (reliability) is effectively defined in terms of process attributes (operational failures), we may also think of this class of prediction systems as using process measures to predict later external product measures.
In an ongoing project, it is possible to validate the estimate by comparing actual values with predicted values. If $E$ is an estimate of a value and $A$ is the actual value, the relative error (RE) in the estimate is:

$$RE = \frac{A - E}{A}$$

Relative error will be negative (if the estimate is greater than the actual value) or positive (if the estimate is less than the actual value).

Inaccuracies can result from either statistical variation or estimating bias.

Estimating inaccuracies must be compensated for throughout the project.
Accuracy of Measurement /2

- *Relative Error (RE)* can be calculated for a collection of estimates, e.g., it is useful to determine if predictions are accurate for a group of projects.

- The **mean relative error (MRE)** is calculated as the sum of *RE* for *n* projects divided by *n*. It is possible for small relative errors to balance out large ones, so the magnitude of the error can also be considered.

- The **mean magnitude of relative error (MMRE)** is calculated using the *absolute value* of *RE*.
MMRE is used to define prediction quality.

On a set of \( n \) projects, if \( k \) is the number of projects whose error \( RE \) is less than or equal to \( q \), then prediction quality is calculated as:

\[
PRED(q) = \frac{k}{n}
\]

**Example:**

\[
PRED(0.25) = 0.47
\]

means that 47% of the predicted values fall within 25% of their actual values.

An estimation technique is (usually) acceptable if

\[
PRED(0.25) = 0.75
\]
Exercise

- Assume you want to predict the error proneness of software modules.

a) How would you assess prediction accuracy?

\[ RE \ (relative \ error) = \frac{(A - E)}{A} \]

with \( A \): actual, \( E \): estimated

\[ MRE = \frac{1}{n} \sum_{i=1}^{n} RE_i \]

\[ MMRE = \frac{1}{n} \sum_{i=1}^{n} |RE_i| \]

b) How would you assess prediction quality?

\[ PRED(q) = \frac{k}{n} \]

where \( k \) is the number of predictions with \(|RE|\leq q\)
Software Measurement: Object Oriented & Subject-Domain Oriented Measurement
Object-Oriented Measurement

- Measurement of a single property is seldom adequate for the purpose of the software measurement because:
  - Direct measurement of a property may be costly, inconvenient, perhaps even impossible. Measured results may be better achieved by observing one or more other properties and deriving the measurement of the required property from them.
  - Measurement is seldom focused on a single property. It may be necessary to characterize a class of objects by the combination of several properties in an object profile.
Definition: Object Profile

- **Profile** is a set of disjoint alternatives called “elements” together with their values or occurrence probabilities.

- **Object Profile** is the set of properties and their values or probabilities of occurrences.

Representing object profile
- Tabular representation
- Graphical representation
Tabular Representation

- Properties 1~n are directly measurable
- Properties n+1~q are measured indirectly
- Such a matrix stores properties which have already been found important, provides information for deriving new indirectly measured properties, and offers a framework for adding new properties when needed.

<table>
<thead>
<tr>
<th>Objects of the model set A</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m₁</td>
</tr>
<tr>
<td>a</td>
<td>m₁(a)</td>
</tr>
<tr>
<td>b</td>
<td>m₁(b)</td>
</tr>
<tr>
<td>c</td>
<td>m₁(c)</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>z</td>
<td>m₁(z)</td>
</tr>
</tbody>
</table>

far@ucalgary.ca

83
Let $A_i = \{a_i, b_i, \ldots, z_i\}$ be a model of $A$ with respect to the $i$th key property.

Then the empirical relational system

$E_i = \{A_i, R_i, O_i\}$

models the set $A$ with respect to the $i$th property, preserving the relations and operations which the property imposes on the objects of $A$.

$S_i = (E_i, F_i, m_i)$ will be a scale for the $i$th property.

Now the set $A$ will be an n-dimensional model of $A$: $A = \{A_1, A_2, \ldots, A_n\}$, each with its own scale.
Assume that $M = \{m_1, m_2, \ldots, m_n\}$ is the set of the real-valued functions on $A$, composed of measures of one of the directly measured key properties.

If one can define a real-valued function $g$ on $A$ in terms of the primitive measures of $M$, then $g$ will be an **indirect measure** of the set of objects of $A$ in terms of the chosen set of key properties, shown by $g(m_1, m_2, \ldots, m_n)$. 
Meaningfulness

- Indirect measures must preserve meaningful empirical properties called *interrelation conditions* which permit one to deduce the meaningful indirect measure as a function of the primitive measures, according to the laws imposed by the model.

- **Example:**
  - Measuring three direct attributes: power (P), voltage (V) and current (I)
  - Measuring power (indirect) in terms of voltage (direct) and current (direct) attributes
  - They should satisfy the condition $P = V \times I$
Example

- Setting failure intensity objective ($\lambda$) in terms of system availability ($A$):

$$A = \frac{1}{1 + \lambda t_m} \quad \text{or} \quad \lambda = \frac{1 - A}{A t_m}$$

$\lambda$ is failure intensity (failure per unit of time)

$t_m$ is downtime per failure

- The representation condition of the indirect measure of $\lambda$ should preserve the relationship.
Scale Invariance

- The scale of any indirect measure is deduced by means of the interrelation condition.

- **Example:** for a database system

\[ \lambda = 20 \text{ /million transactions} \]
\[ t_m = 6 \text{ minutes} \]

\( \lambda \) and \( t_m \) must have a compatible scale to calculate \( A \)

If there are 20,000 transactions/day and a day has 8 working hours, then \( t_m \) can be expressed as equivalent to 250 transactions and

\[ A = \frac{1}{1 + \lambda t_m} = \frac{1}{1 + 0.005} = 0.995 \]
When approaching the issue of object-oriented measurement, it was necessary to broaden the perspective from a single property to the many-featured characterization of objects.

When considering subject domain-oriented measurement, one must widen ones horizons once more, to consider the problem of characterizing by measurement any of the objects of a discipline.
The international measurement system (SI). SI provides measures for characterizing observable entities of the physical universe by means of only seven orthogonal base units:

1. length (metre)
2. mass (kilogram)
3. time (second)
4. temperature (degrees Kelvin)
5. electric current (ampere)
6. luminous intensity (candela)
7. the amount of substance (mole)

Other physical quantities are measured in units derived from these, such as velocity measured in metre/second (m/s).
At present, software engineering is still in need of stronger discipline-wide conceptual foundations and more comprehensive formal theories and empirical laws.

The subject domain is not ready for the formulation of a comprehensive measurement system.

However, there is widespread recognition of the need for measurement-based characterization of software, stemming from:

- Discontent about the quality of software and the productivity of the industry.
- Anxiety about the safety of software-related systems.
- Concern about the security of data entrusted to software.
- Convincing evidence of safety and security.
Software Measurement!
References


References