Software Metrics
Chapter-2

The Basics of Measurement

The Representational Theory Of Measurement

- Developed as a classical discipline from the physical sciences
- Provides rules for:
    - Making consistent measurements
    - Interpreting data resulting from measurement
- Measurement theory tells us the rules, laying the groundwork for developing and reasoning about all kinds of measurement.

- Empirical Relations:
- The representational theory of measurement seeks to formalize our intuition about the way the world works.
- Data obtained as measures should represent attributes of observed entities
- Manipulating data should preserve observed relationships

  E.g. Taller than
- Binary relation defined on the set of pairs of people Either

  A is taller than B, or

  B is taller than A

- Empirical relations are not restricted to binary relations
- They can be unary (e.g., A is tall), ternary (A sitting on B's shoulders is taller than C), etc.
- Empirical relations are mappings from the empirical, real world to a formal mathematical world.

E.g.

	More Functionality				More User-Friendly			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**
A	—	80	10	80	—	45	50	44
B	20	—	5	50	55	—	52	50
C	90	95	—	96	50	48	—	51
D	20	50	4	—	54	50	49	—

- Sampling 100 Users to Express Preferences among Products A, B, C, and D on the above table
- Empirical relation: *greater functionality than*
- Mapping: program *x* has greater functionality than program *y* if the survey result for cell (*x*,*y*) exceeds 60%.

- So we can define
- **Measurement as** – a mapping from the empirical world to the formal, relational world.

and

- **Measure as** – number or symbol assigned to an entity by the mapping in order to characterize an attribute.

Rules of Mapping

- Measures must specify domain and range as well as the rule for performing the mapping
  - Domain – real world is domain of mapping that defines the measurement
  - Range – the mathematical world into which real-world attributes are mapped

- Examples
  - Measuring height:
    - Is height measured in inches, centimeters, feet?
    - Are people measured sitting or standing?
    - Are shoes allowed to be worn during the measurement?
  - Measuring lines of code
    - Are non-executable lines counted?
      - Declarations
      - Compiler Directives
      - Comments
      - Blank lines

The Representational Condition of Measurement

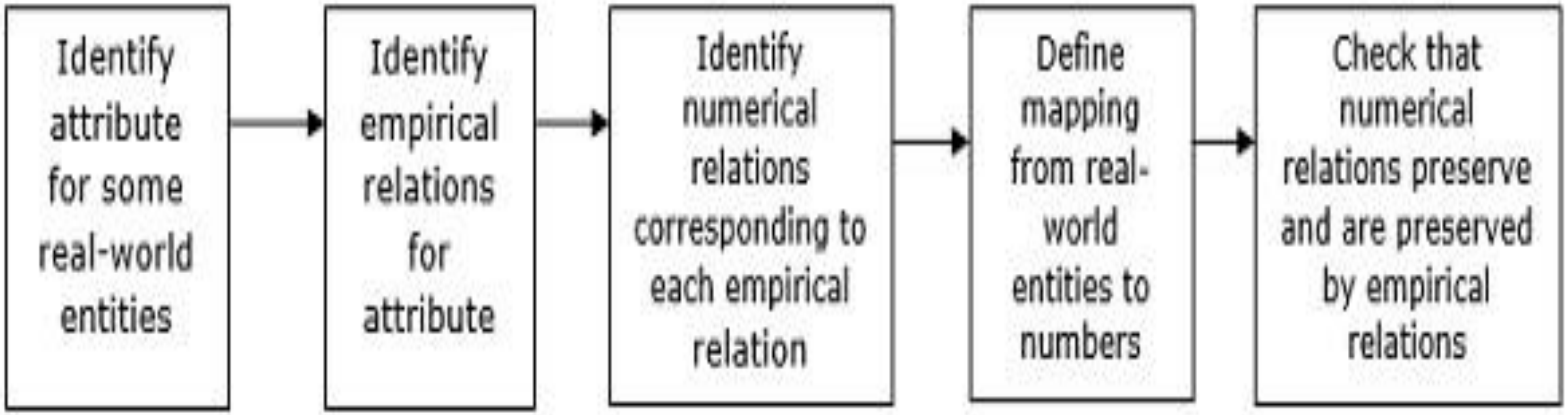
- The representational condition states that a measurement mapping **(M)** must map entities into numbers, and empirical relations into numerical relations in such a way that the empirical relations preserve and are preserved by numerical relations.
- E.g1. The empirical relation 'taller than' is mapped to the numerical relation ">".i.e., **X is taller than Y, if and only if M(X) > M(Y)**
- Since, there can be many relations on a given set, the representational condition also has implications for each of these relations.
- For the unary relation 'is tall', we might have the numerical relation **X > 50**
- The representational condition requires that for any measure **M**, **X is tall if and only if M(X) > 50**

- E.g2: Software failures criticality
- Three types of failures examined:
    - Delayed response
    - Incorrect output
    - Data loss
- At this point, we have a relation system consisting of 3 unary relations
    - R1 for delayed response
    - R2 for incorrect output
    - R3 for data loss

- With above information, we can’t yet judge the relative criticality of these types of failures.
- We can find a representation in the set of real numbers by choosing three distinct numbers:
  - M(delayed response) = 6
  - M(incorrect output)=4
  - M(data loss)=50

- Further investigation of criticality reveals that data loss is more critical than incorrect output, which in turn is more critical than a delayed response.
- To develop a real-number representation for this enriched relation, we must be more careful in assigning numbers.
- Using “>” to mean “more critical than”, data-loss failures must be mapped to a higher number than incorrect output failures, which in turn must mapped to a higher number than delayed responses.

Key Stages of Formal Measurement



Measurement and models

- a *model* is an abstraction of reality, allowing us to strip away detail and view an entity or concept from a particular perspective
- Models come in many different forms: as equations, mappings, or diagrams.

Direct and Derived Measurement

- *Direct measurement* of an attribute of an entity involves no other attribute or entity.

  E.g. length

- On the other hand, when measure one physical object by computing value of another objects then the measurement is derived measurement.

  E.g. density of one physical object is derived form mass and volume

  as

  density =mass/volume

- direct measures are commonly used in sofware engineering:
  - *Size* of source code (measured by LOC)
  - *Schedule* of the testing process (measured by elapsed time in hours)
  - *Number of defects discovered* (measured by counting defects)
  - *Time* a programmer spends on a project (measured by months worked)

- Common Derived Measures Used in Software Engineering

$$Programmer\ Productivity = \frac{LOC\ produced}{Person\ months\ of\ effort}$$

$$Module\ Defect\ Density = \frac{Number\ of\ defects}{Module\ size}$$

$$Defect\ Detection\ Efficiency = \frac{Number\ of\ defects\ detected}{Total\ number\ of\ defects}$$

$$Requirement\ Stability = \frac{Number\ of\ initial\ requirements}{Total\ number\ of\ requirements}$$

$$Test\ Effectiveness\ Ratio = \frac{Number\ of\ items\ covered}{Total\ number\ of\ items}$$

$$System\ spoilage = \frac{Effort\ spent\ for\ fixing\ faults}{Total\ project\ effort}$$

Measurement for Prediction

- For allocating the appropriate resources to the project, we need to predict the effort, time, and cost for developing the project.
- The measurement for prediction always requires a mathematical model that relates the attributes to be predicted to some other attribute that we can measure now.
- Hence, a prediction system consists of a mathematical model together with a set of prediction procedures for determining the unknown parameters and interpreting the results.

Measurement Scales and Scale Types

- Nominal
- Ordinal
- Interval
- Ratio
- Absolute

- Nominal scale type:
  - This is the most primitive form of measurement.
  - Has two major characteristics:
    - 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.
    - E.g.

$$M_1(x) = \begin{cases} 1 \text{ if } x \text{ is specification fault} \\ 2 \text{ if } x \text{ is design fault} \\ 3 \text{ if } x \text{ is code fault} \end{cases}$$

$$M_2(x) = \begin{cases} 101 \text{ if } x \text{ is specification fault} \\ 2.73 \text{ if } x \text{ is design fault} \\ 69 \text{ if } x \text{ is code fault} \end{cases}$$

Ordinal scale

- Augments nominal scale with ordering information.
- Three major characteristics
  - Empirical relation system consists of classes that are ordered with respect to the attribute
  - Any mapping preserving the ordering (i.e., a monotonic function) is acceptable
  - Numbers represent ranking only, so arithmetic operations have no meaning
- Set of admissible transformations is set of all monotonic mappings
- Example – software “complexity” – two valid measures

Value	Meaning
1	Trivial
2	Simple
3	Moderate
4	Complex
5	Incomprehensible

Value	Meaning
2	Trivial
4	Simple
6	Moderate
9	Complex
12	Incomprehensible

$$M_4(x) = \begin{cases} 1 \text{ if } x \text{ is trivial} \\ 1 \text{ if } x \text{ is simple} \\ 3 \text{ if } x \text{ is moderate} \\ 4 \text{ if } x \text{ is complex} \\ 5 \text{ if } x \text{ is incomprehensible} \end{cases} \qquad M_5(x) = \begin{cases} 1 \text{ if } x \text{ is trivial} \\ 3 \text{ if } x \text{ is simple} \\ 2 \text{ if } x \text{ is moderate} \\ 4 \text{ if } x \text{ is complex} \\ 10 \text{ if } x \text{ is incomprehensible} \end{cases}$$

Interval scale

- Captures information about size of intervals that separate classes.
- Three characteristics
  - 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.

A *ratio scale*

- A *ratio scale* has the following characteristics:
  - It is a measurement mapping that preserves ordering, the size of intervals between entities, and 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.

Absolute scale

- Most restrictive in terms of admissible transformations
- For any two measures, M and M’, there’s only one admissible transformation (identity transformation), since there’s only one way to make the measurement.
- 4 characteristics
  - Measurement is made simply by counting the number of elements in the entity set
  - Attribute always takes the form of “number of occurrences of x in the entity”
  - Only one possible measurement mapping, namely the actual count
  - All arithmetic analysis of the resulting count is meaningful.
- Example – lines of code in a module is an absolute scale measure.

To summarize

Scale type	Admissible transformations	Examples
Nominal	1-1 mapping	Labeling, classifying entities
Ordinal	Monotonic increasing function	Preference, hardness, air quality, intelligence tests (raw scores)
Interval	$M' = aM + b$, $a > 0$	Relative time, temperature (Fahrenheit, Celsius), intelligence tests (standardized scores)
Ratio	$M' = aM$, $a > 0$	Time interval, length, temperature (Kelvin)
Absolute	$M' = M$	Counting entities

MEANINGFULNESS IN MEASUREMENT

- After making measurements, key question is “can we deduce meaningful statements about entities being measured?”
- Harder to answer than it first appears – consider these statements:
  - The number of errors discovered during the integration testing of a program X was at least 100
  - The cost of fixing each error in program X 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

  the statements in **red** are meaningless. Why? Please go through the textbook