The Basics of Measurement

SE-843: SOFTWARE METRICS

Empirical Relations

The *representational theory of measurement* seeks to formalize our intuition about the way the world works.

In other words, *taller than* is a binary relation defined on the set of pairs of people. Given any two people, *x* and *y*, we can observe that

- ✓ *x* is taller than *y*, or
- ✓ *y* is taller than *x*

Therefore, we say that *taller than* is an empirical relation for height.

we define measurement as the mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute.

Measurement Scale

Likert Scale

Give the respondent a statement with which to agree or disagree. Example:

This software program is reliable.

Strongly Agree	Agree	Neither agree nor disagree	Disagree	Strongly Disagree

Measurement Scale

Forced Ranking

Give *n* alternatives, ordered from 1 (best) to *n* (worst). Example:

Rank the following five software modules in order of maintenance difficulty, with 1 = least complex, 5 = most complex:

—	Module A
—	Module B
—	Module C
—	Module D
—	Module E

Measurement Scale

Verbal Frequency Scale

Example: How often does this program fail?

Always	Often	Sometimes	Seldom	Never

Measurement Scale

Ordinal Scale

List several ordered alternatives and have respondents select one. For example:

How often does the software fail?

1. Hourly
2. Daily
3. Weekly
4. Monthly
5. Several times a year
6. Once or twice a year
7. Never

Measurement Scale

Comparative Scale

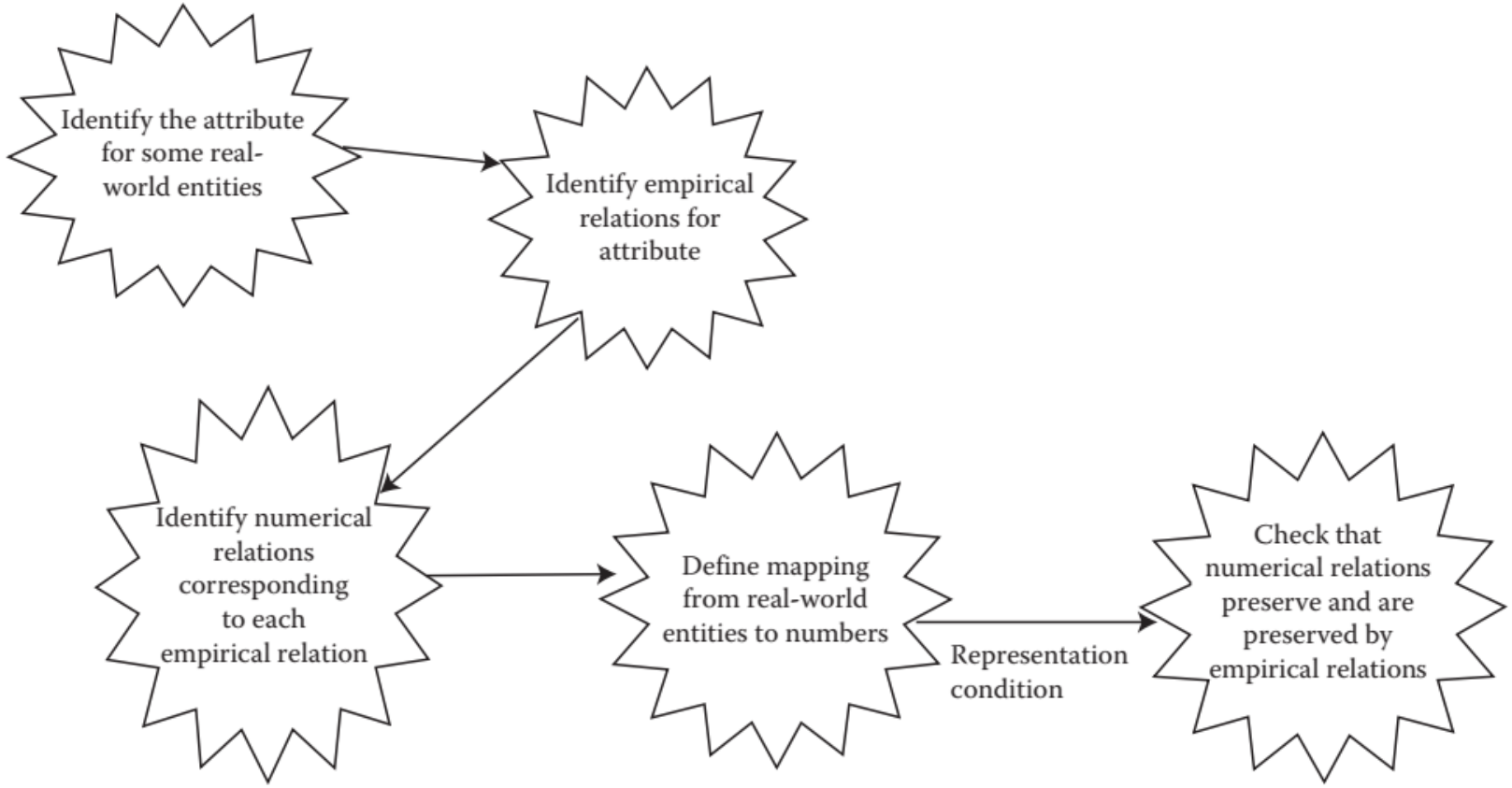
Very superior			About the same				Very inferior
1	2	3	4	5	6	7	8

Measurement Scale

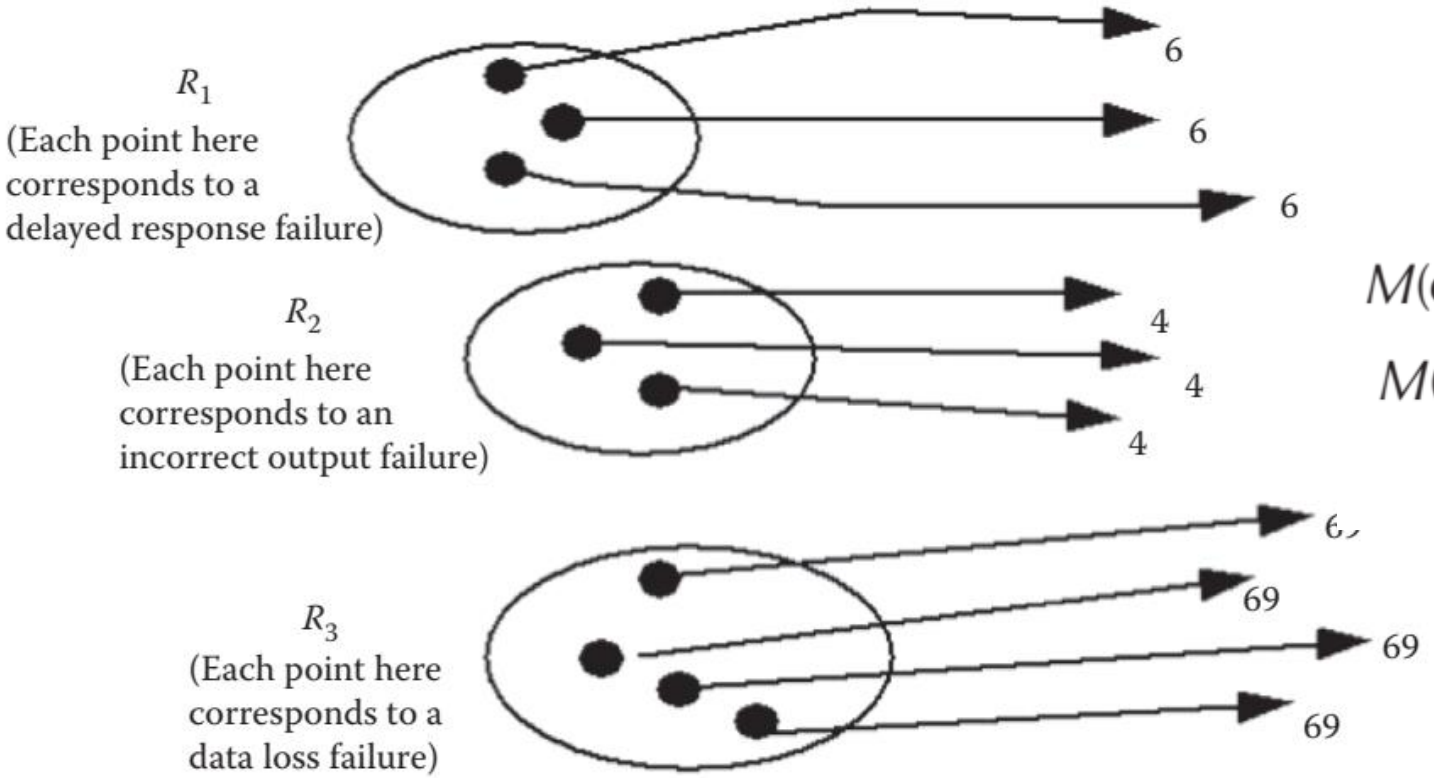
Numerical Scale

Unimportant							Important
1	2	3	4	5	6	7	8

The Representation Condition of Measurement



Measurement Mapping



- Delayed-response
- Incorrect output
- Data-loss

M(each delayed response) = 6

M(each incorrect output) = 4

M(each data loss) = 69

Specific Measurement in Software

	Entity	Attribute	Measure
1	Completed project	Duration	Months from start to finish
2	Completed project	Duration	Days from start to finish
3	Program code	Length	Number of lines of code (LOC)
4	Program code	Length	Number of executable statements
5	Integration testing process	Duration	Hours from start to finish
6	Integration testing process	Rate at which faults are found	Number of faults found per KLOC (thousand LOC)
7	Test set	Efficiency	Number of faults found per number of test cases
8	Test set	Effectiveness	Number of faults found per KLOC (thousand LOC)
9	Program code	Reliability	Mean time to failure (MTTF) in CPU hours
10	Program code	Reliability	Rate of occurrence of failures (ROCOF) in CPU hours

Direct and Derived Measurement

Direct measurement of an attribute of an entity involves no other attribute or entity. For example, *length* of a physical object can be measured without reference to any other object or attribute. E.g. *Size* of source code (measured by LOC), *Number of defects discovered*, etc.

Derived measurement is often useful in making visible the interactions between direct measurements. That is, it is sometimes easier to see what is happening on a project by using combinations of measures. E.g. fault density (i.e., the derived measure defined as faults per KLOC).

Examples of Common Derived Measures Used in Software Engineering

Programmer productivity	LOC produced/person-months of effort
Module defect density	Number of defects/module size
Defect detection efficiency	Number of defects detected/total number of defects
Requirements stability	Number of initial requirements/total number of requirements
Test coverage	Number of test requirements covered/total number of test requirements
System spoilage	Effort spent fixing faults/total project effort

Measurement scales and scale types

1. Nominal
2. Ordinal
3. Interval
4. Ratio
5. Absolute

Nominal Scale Type

Suppose we define classes or categories, and then place each entity in a particular class or category, based on the value of the attribute. This categorization is the basis for the most primitive form of measurement, the nominal scale. Thus, the nominal scale has two major characteristics:

1. The empirical relation system consists only of different classes; there is no notion of ordering among the classes.
2. Any distinct numbering or symbolic representation of the classes is an acceptable measure

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

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

Ordinal Scale Type

We can often augment the nominal scale with information about an ordering of the classes or categories creating an ordinal scale. The ordering leads to analysis not possible with nominal measures. The ordinal scale has the following characteristics:

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

Ordinal Scale Type

$$M_1(x) = \begin{cases} 1 \text{ if } x \text{ is trivial} \\ 2 \text{ if } x \text{ is simple} \\ 3 \text{ if } x \text{ is moderate} \\ 4 \text{ if } x \text{ is complex} \\ 5 \text{ if } x \text{ is incomprehensible} \end{cases} \qquad M_2(x) = \begin{cases} 1 \text{ if } x \text{ is trivial} \\ 2 \text{ if } x \text{ is simple} \\ 3 \text{ if } x \text{ is moderate} \\ 4 \text{ if } x \text{ is complex} \\ 10 \text{ if } x \text{ is incomprehensible} \end{cases}$$

$$M_3(x) = \begin{cases} 0.1 \text{ if } x \text{ is trivial} \\ 1001 \text{ if } x \text{ is simple} \\ 1002 \text{ if } x \text{ is moderate} \\ 4570 \text{ if } x \text{ is complex} \\ 4573 \text{ if } x \text{ is incomprehensible} \end{cases}$$

However, neither M_4 nor M_5 is valid:

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

Interval Scale Type

The interval scale carries more information still, making it more powerful than nominal or ordinal. This scale 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.

- 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 Type

We can measure air temperature on a Fahrenheit or Celsius scale. Thus, we may say that it is usually 20° Celsius on a summer's day in London, while it may be 30° Celsius on the same day in Washington, DC. The interval from one degree to another is the same, and we consider each degree to be a class related to heat. That is, moving from 20° to 21° in London increases the heat in the same way that moving from 30° to 31° does in Washington. However, we cannot say that it is two-third as hot in London as Washington; neither can we say that it is 50% hotter in Washington than in London. Similarly, we cannot say that a 90° Fahrenheit day in Washington is twice as hot as a 45° Fahrenheit day in London.

Ratio Scale Type

Although the interval scale gives us more information and allows more analysis than either nominal or ordinal, we sometimes need to be able to do even more. For example, 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 need for ratios gives rise to the ratio scale, the most useful scale of measurement, and one that is common in the physical sciences. A *ratio scale* has the following characteristics:

- It is a measurement mapping that preserves ordering
- There is a zero element, representing total lack of the attribute.
- Measurement mapping must start at zero and increase at equal intervals, (units)

Example Ratio Scale Type

The length of software code is also measurable on a ratio scale. As with other physical objects, we have empirical relations like *twice as long*. The notion of a zero-length object exists—an empty piece of code. We can measure program length in a variety of ways, including LOC, thousands of LOC, the number of characters contained in the program, the number of executable statements, and more. Suppose M is the measure of program length in LOC, while M' captures length as number of characters. Then we can transform one to the other by computing $M' = aM$, where a is the average number of characters per line of code.

Absolute Scale Type

As the scales of measurement carry more information, the defining classes of admissible transformations have become increasingly restrictive. The absolute scale is the most restrictive of all. For any two measures, *M* and *M′*, As the scales of measurement carry more information, the defining classes of admissible transformations have become increasingly restrictive. The absolute scale is the most restrictive of all. For any two measures, *M* and *M′*,

- 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, namely the actual count, and there is only one way to count elements

Scales of Measurement

Scale Type	Admissible Transformations (How Measures M and M' must be Related)	Examples
Nominal	1–1 mapping from M to M'	Labeling, classifying entities
Ordinal	Monotonic increasing function from M to M', that is, $M(x)\ M(y)$ implies $M'(x)\ M'(y)$	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

Statistical Operations on Measures

We have measured an attribute for 13 entities, and the resulting data points in ranked order are:
2, 2, 4, 5, 5, 8, 8, 10,11, 11, 11, 15, 16

- The *mean* of this set of data (i.e., the sum divided by the number of items) is *8.3.*
- The *median* (i.e., the value of the middle-ranked item) is *8.*
- The *mode* (i.e., the value of the most commonly occurring item) is *11.*

Summary of Measurement Scales and Statistics

Scale Type	Defining Relations	Examples of Appropriate Statistics	Appropriate Statistical Tests
Nominal	Equivalence	Mode Frequency	Nonparametric
Ordinal	Equivalence Greater than	Median Percentile Spearman r_S Kendall τ Kendall W	Nonparametric
Interval	Equivalence Greater than Known ratio of any intervals	Mean Standard deviation Pearson product-moment correlation Multiple product-moment correlation	Non-parametric
Ratio	Equivalence Greater than Known ratio of any intervals Known ratio of any two scale values	Geometric mean Coefficient of variation	Nonparametric and parametric

End of Chapter 2