Software Metrics (PA1407)

Lecture 2

The basics of measurement

Basics of measurement

- Questions that are relatively easy to answer for non-software entities are difficult for software.
  - Do we know enough about "complexity" of programs to be able to measure it?
  - Does a count of the number of "bugs" found in a system during integration testing measure the quality of the system? If not, what does the count tell us?
  - What meaningful statements can we make about an attribute and the entities that possess it?
    - For instance, is it meaningful to talk about doubling a design's quality? If not, how do we compare two different designs?
  - What meaningful operations can we perform on measures?
    - For instance, is it sensible to compute average productivity for a group of developers?
- To answer these questions, we must establish the basics of a theory of measurement.

Representational theory of measurement

- In any measurement activity, there are rules to be followed.
  - Important for consistent measurement, and basis for interpretation.
- 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.
    - Manipulation of the collected data should preserve relationships that we observe among the entities.
  - Example : Height
  - Preservation of intuition and observation

Representational theory of measurement

- Measurement is a mapping from empirical world to the formal, relational world.
  - A measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute.
- Rules of the mapping
  - The real world is domain of the mapping, and mathematical world is the range.
  - When we map the attribute to a mathematical system, we may have many choices for the mapping and range. We can use real numbers, integers or non-numeric symbols.
    - LOC example as size measure

Rules of the mapping: Example

Statement type	*Include?*	*Exclude?*
Executable		
Non-executable		
Declarations		
Compiler directives		
Comments		
On their own lines		
On lines with source code		
Banners and non-blank spacers		
Blank (empty) comments		
Blank lines		

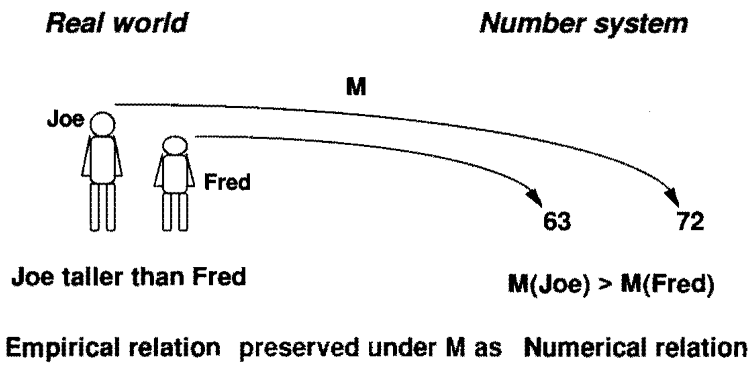
How produced	*Include?*	*Exclude?*
Programmed		
Generated with source code generators		
Converted with automatic translators		
Copied or reused without change		
Modified		
Removed		

Origin	*Include?*	*Exclude?*
New work: no prior existence		
Prior work: taken or adapted from		
A previous version, build or release		
Commercial, off-the-shelf software, other than libraries		
Government furnished software, other than reuse libraries		
Another product		
A vendor-supplied language support library (unmodified)		
A vendor-supplied operating system or utility (unmodified)		
A local or modified language support library or operating system		
Other commercial library		
A reuse library (software designed for reuse)		
Other software component or library		

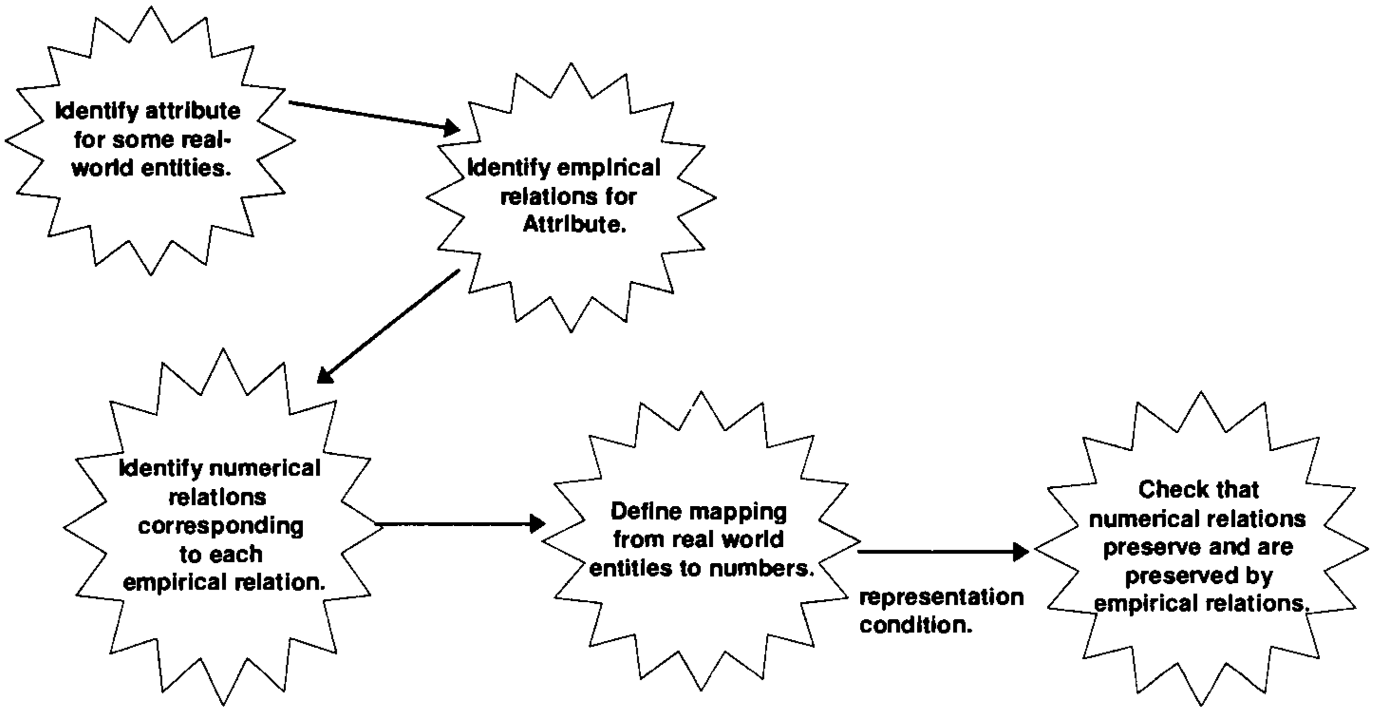
Figure 2.3: Adapted from portion of US Software Engineering Institute checklist for lines-of-code count

Representational theory of measurement

- Representation condition of measurement
  - The behavior of the measures in the number system should be the same as the corresponding elements in the real world
  - Height Example:
    - A is taller than B if and only if M(A)>M(B)



Key stages of formal measurement



Examples of measures in SE

- How good a measure is **faults per KLOC**?
  - The answer depends entirely on the entity-attribute pair connected by the mapping
  - Intuitively, faults per KLOC is a good measure of the rate at which faults are found for the testing process (example 6).
  - However, it may not be such a good measure of efficiency of the tester (example 7). Why?

	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	Tester	Efficiency	Number of faults found per KLOC (thousand LOC)
8	Program code	Quality	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

Defining attributes

- When measuring, there is always a danger that we focus too much on the formal, mathematical system, and not enough on the empirical one.
- We rush to create mappings and then manipulate numbers, without giving careful thought to the relationships among entities and their attributes in the real world
  - Example: The cyclomatic number presents only a partial view of complexity.

Direct and indirect measurement

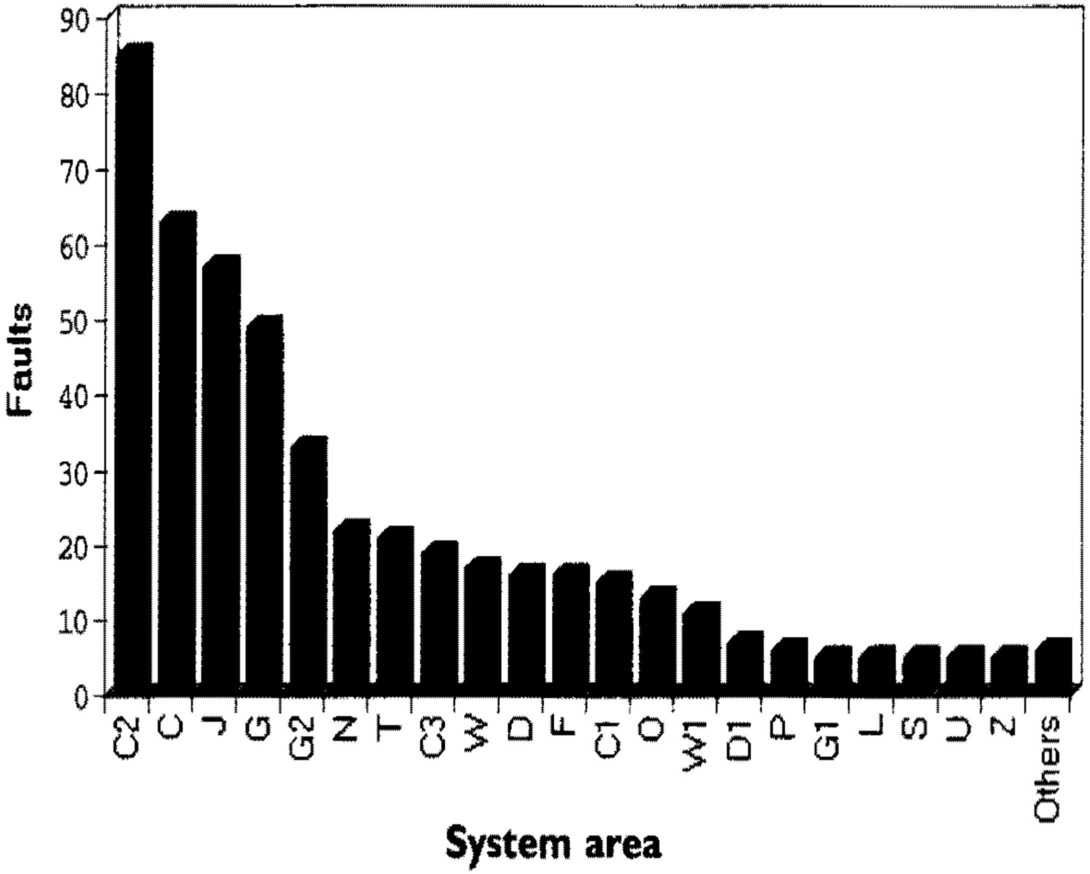
- Direct measurement of an attribute of an entity involves no other attribute or entity.
  - Distance travelled and time taken are direct measures, while speed is calculated using both distance and time, and hence is indirect.

Table 2.4: Examples of common indirect measures used in software engineering

Programmer productivity	$\frac{\text{LOC produced}}{\text{person months of effort}}$
Module defect density	$\frac{\text{number of defects}}{\text{module size}}$
Defect detection efficiency	$\frac{\text{number of defects detected}}{\text{total number of defects}}$
Requirements stability	$\frac{\text{number of initial requirements}}{\text{total number of requirements}}$
Test effectiveness ratio	$\frac{\text{number of items covered}}{\text{total number of items}}$
System spoilage	$\frac{\text{effort spent fixing faults}}{\text{total project effort}}$

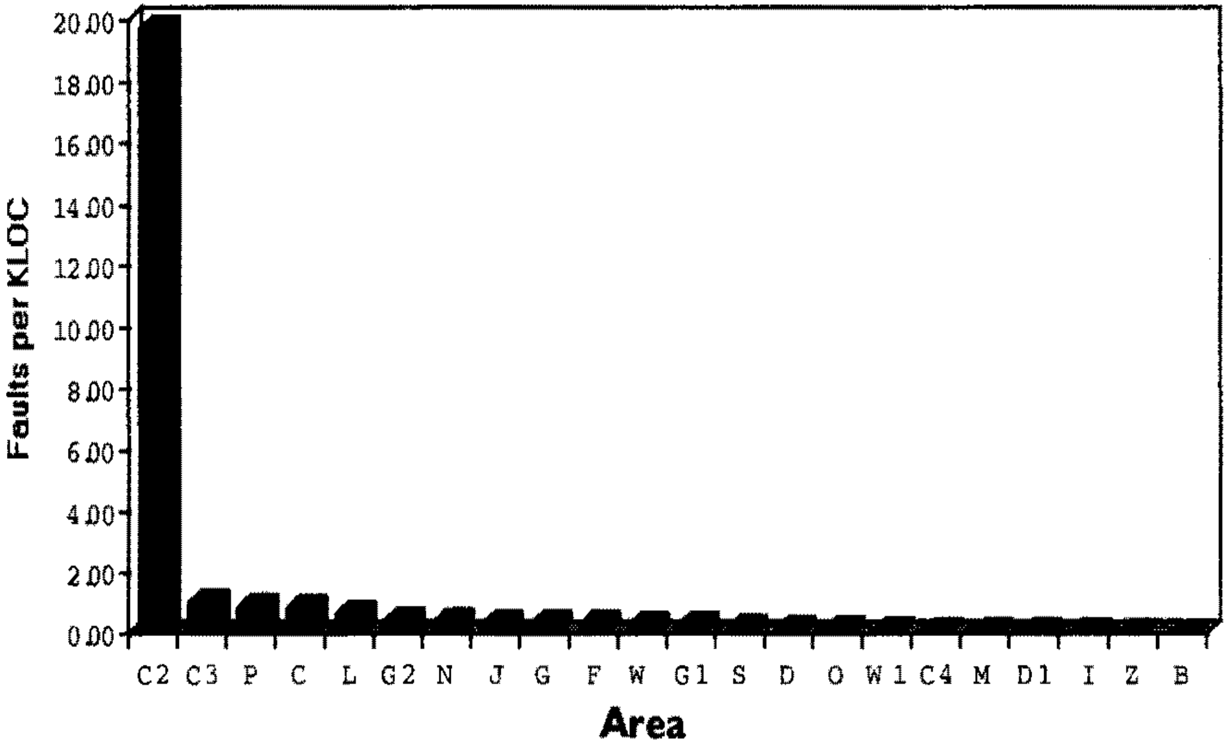
Direct measurement example

It appears as if there are five system areas that contain the most problems for the developers maintaining this system.



Indirect measurement example

From the indirect measurement, it is very clear that one system area is responsible for the majority of the problems.
In fact, system area C2 is only 4000 lines of code out of two million, but it is a big headache for the maintainers.



Measurement for prediction

- Measurement for assessment
  - Measuring product size and structure
- Measurement for prediction
  - Predicting product maintainability
- A **prediction system** consists of a mathematical model together with a set of prediction procedures for determining unknown parameters and interpreting results (Littlewood, 1988).

Scale types

- There are different types of measurement scales.
- Five major types
  - nominal
  - ordinal
  - interval
  - ratio
  - Absolute
- Knowing the characteristics of each type helps us to
  - Select most suitable one for measuring an attribute of interest.
  - interpret the measures

Nominal Scale

- The most primitive form of measurement
- It defines classes or categories, and places entities in a particular class or category, based on the value of the attribute.
- Characteristics
  - Entities are only organized in different classes.
  - The empirical relation system consists only of different classes; No notion of ordering among the classes
  - Any numbering or symbolic representation of the classes is acceptable, but there is no notion of magnitude associated with the numbers of symbols.
- Examples ??
  - Categorizing people based on gender or job role
  - Categorizing testing tools based on license types
  - Categorizing effort predictors based on their type

Ordinal Scale

- The ordinal scale is often useful to augment the nominal scale with information about an ordering of classes or categories.
- Characteristics
  - The empirical relation system consists of classes that are ordered with respect to the attribute.
  - Any mapping that preserves the ordering is acceptable.
  - The numbers represent ranking only, so addition, subtraction and other arithmetic operations have no meaning.

Ordinal scale example

- Suppose our entities are software modules, and the attribute is “complexity”. We may be able to define five distinct classes of module complexity:
  - "trivial," "simple," "moderate," "complex," and "incomprehensible.”
  - There is an implicit order relation of “less complex than” on these classes.
  - We cannot be as free in our choice of mapping as we could with a nominal measure
  - *Mapping function M* must be a monotonically increasing function
- More examples ??

Interval scale

- This scale captures more information than ordinal and nominal scales.
  - Besides order, it also captures information about the size of the jump from one class to another.
- Characteristics
  - An interval scale preserves order, as with an ordinal scale
  - An interval scale preserves differences but not ratios.
  - Addition and subtraction are acceptable operations, but **not** multiplication and division
- Examples: Very few in SE
  - Project elapsed time (Entity: project, Attribute: elapsed time)
  - Modules example given with ordinal scale provided that
    - Difference in complexity between a trivial and simple system is the same as that between a simple and moderate system and so on.
  - Air temperature

Ratio scale

- The need for ratios give rise to the ratio scale, the most useful scale of measurement.
- 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.
- The zero element with length of physical objects example
  - Zero element is theoretical, in the sense that we can think of an object as having no length at all; thus, the zero-length object exists as a limit of things that get smaller and smaller

Ratio scale examples

- Length of software code
  - Entity: code
  - Attribute: length
  - Measure: LOC
- Development effort of a project
  - Entity: project
  - Attribute: effort
  - Measure: hours

Absolute scale

- Absolute scale is the most restrictive of all.
- Characteristics
  - 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.
  - All arithmetic analysis of the resulting count is meaningful.
- Examples
  - Number of bugs
  - Number of developers working in a project.

Summary for scale types

Scale type	Examples
Nominal	Labelling, classifying entities
Ordinal	Preference, hardness, air quality, intelligent tests (raw scores)
Interval	Relative time, temperature (Celsius, Fahrenheit), intelligence tests (standardised scores)
Ratio	Time interval, length, temperature (Kelvin)
Absolute	Counting entities

Statistical operations on measures

Table 2.8: Summary of measurement scales and statistics relevant to each (Siegel and Castellan, 1988)

Scale type	Defining relations	Examples of appropriate statistics	Appropriate statistical tests
Nominal	Equivalence	Mode Frequency	Non-parametric
Ordinal	Equivalence Greater than	Median Percentile Spearman r Kendall τ Kendall W	Non-parametric
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	Non-parametric and parametric

Acknowledgement

- Lecture notes are prepared from following source:
  - T1: Software Metrics - A Rigorous & Practical Approach, 2nd edition, Authors: N. E. Fenton, S. L. Pfleeger, Publishers: International Thomson Computer Press, 1996, ISBN: 1-85032-275-9.