Software Metrics

Lecture 2
Measurement Theory

Yuming Zhou

What is Measurement?

- **Measurement** is the process by which *numbers* or *symbols* are assigned to *attributes* of *entities* in the real world in such a way as to ascribe them according to *defined rules*

$$entity = \left\{ \begin{array}{ll} attribute_1 & (value_{11}, value_{12}, \ldots) \\ attribute_2 & (value_{21}, value_{22}, \ldots) \\ \ldots & \ldots \\ attribute_n & (value_{n1}, value_{n2}, \ldots) \end{array} \right\}$$

What are Metrics and Software Metrics?

- **Metrics** are standards (i.e., commonly accepted scales) that define measurable attributes of entities, their units and their scopes

- **Software metrics** are metrics that are used to quantify:
  - Software product
  - Software development resources
  - Software development process

Why Use Software Metrics?

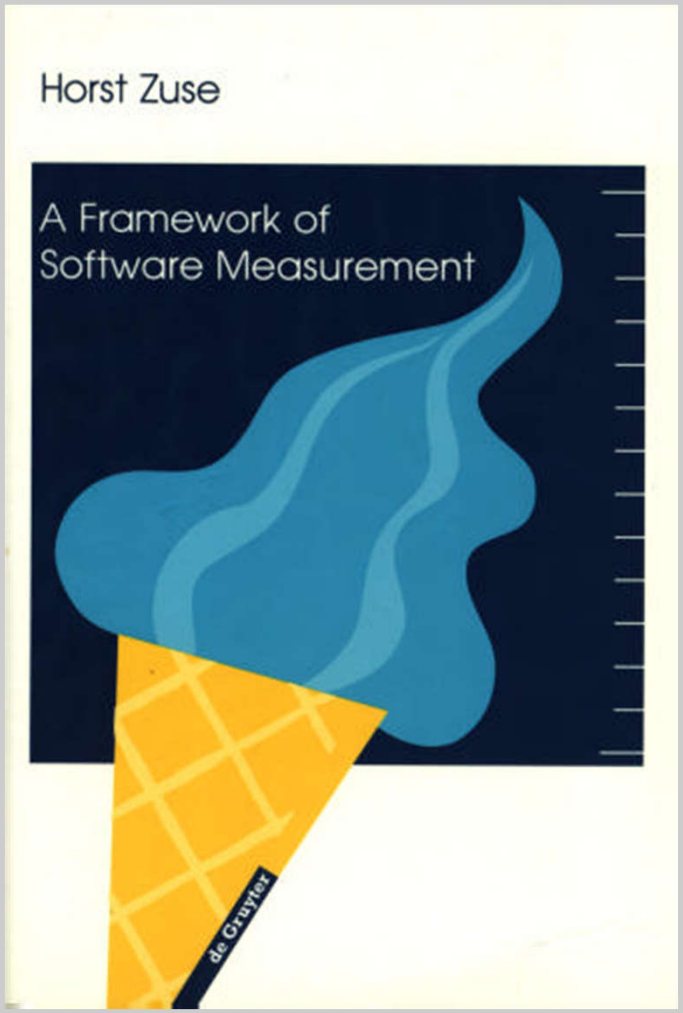
- We use them to derive a basis for estimation
- To track project progress
- To determine complexity
- To help us to understand when we have achieved a desired state of software quality
- To analyze our defects

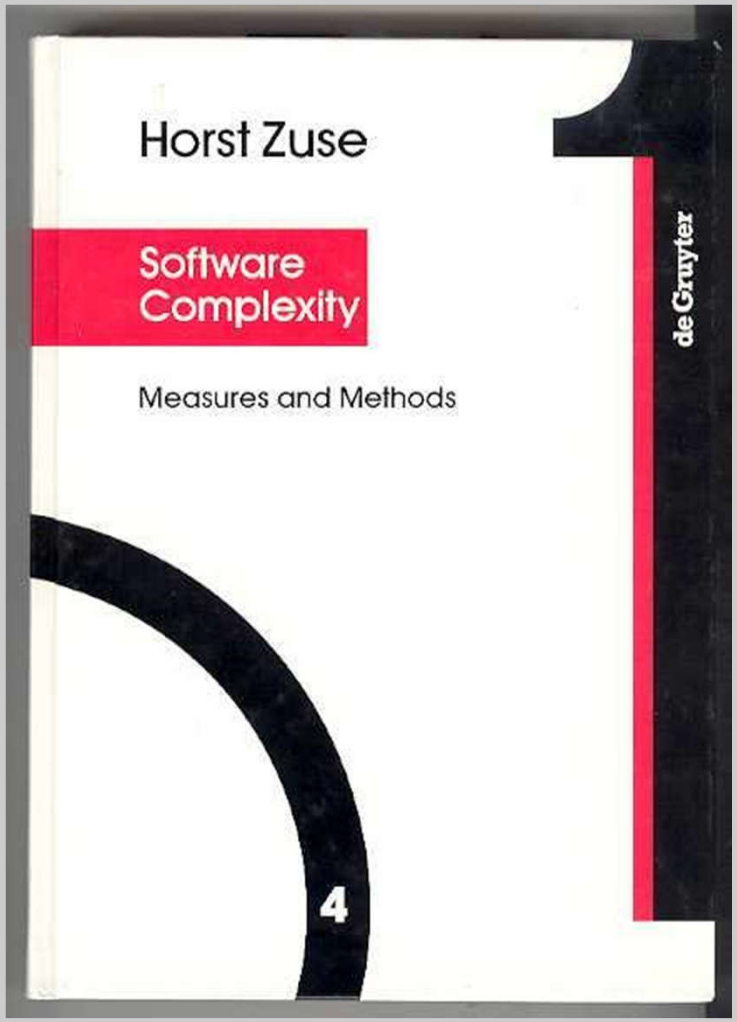
They help us to understand, control and improve software!!!

什么是一个“好”的软件度量

- A. Meneely, et al. Validating software metrics: a spectrum of philosophies. ACM TOSEM 2012
- L. Briand, et al. Property-based software engineering measurement. IEEE TSE 1996
- B. Kitchenham, et al. Towards a framework for software measurement validation. IEEE TSE 1995
- N. Fenton. Software measurement: a necessary scientific basis. IEEE TSE 1994
- N. Schneidewind. Methodology for validating software metrics. IEEE TSE 1992
- E. Weyuker. Evaluating software complexity measures. IEEE TSE 1988

什么是一个“好”的软件度量





Contents

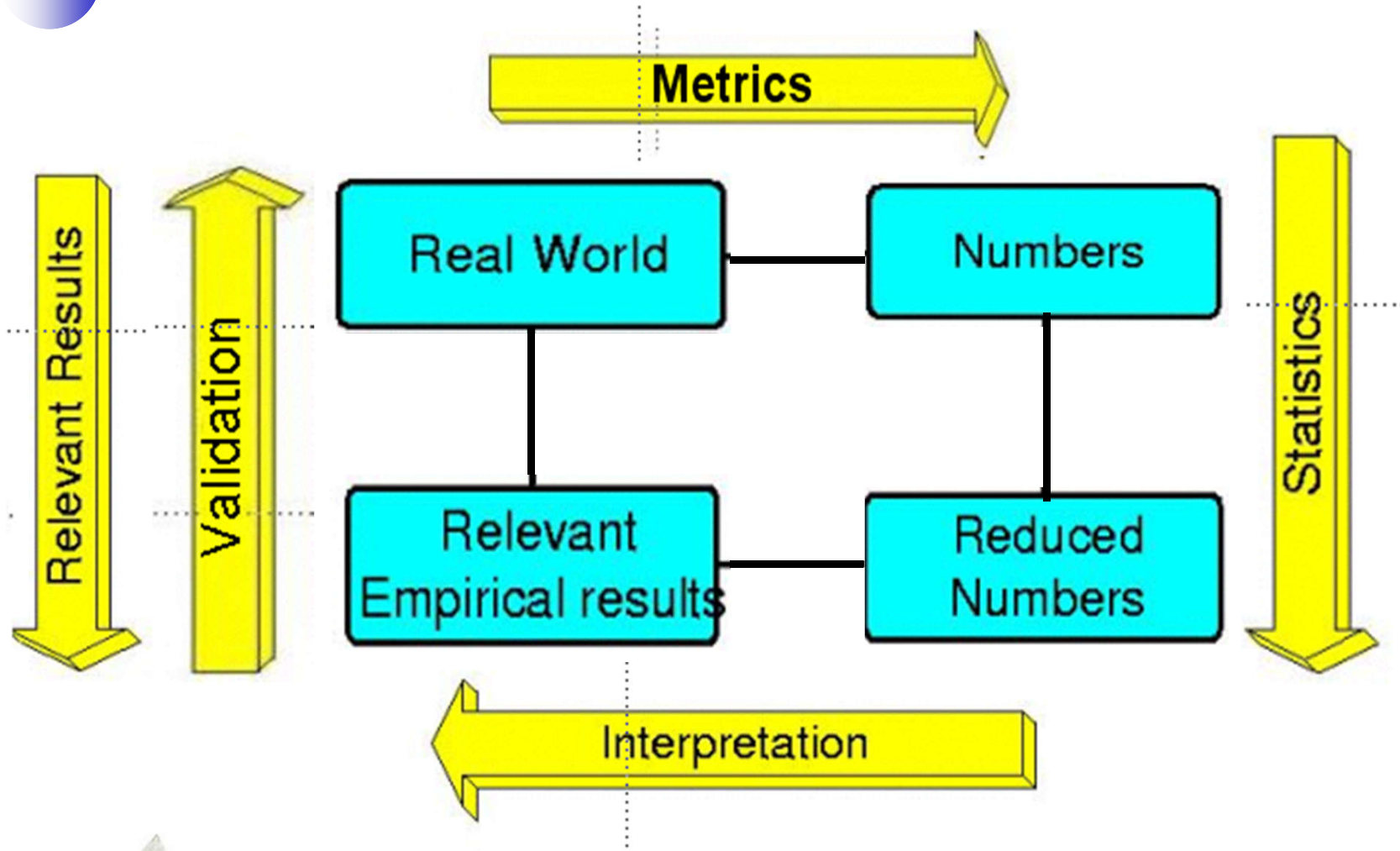
- The Process of Measurement
- Measurement Scales
- Scale Types
- Meaningfulness in Measurement

Section 1

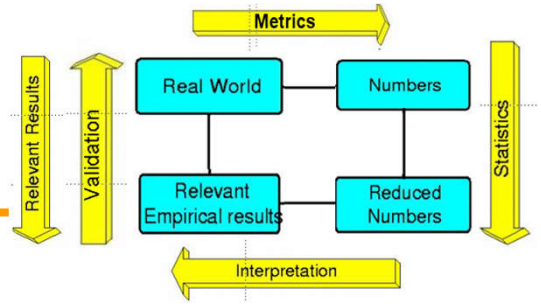
The Process of Measurement

- **Measurement Process**
- **Key Problems**
- **Key Stages of Measurement**

Measurement Process



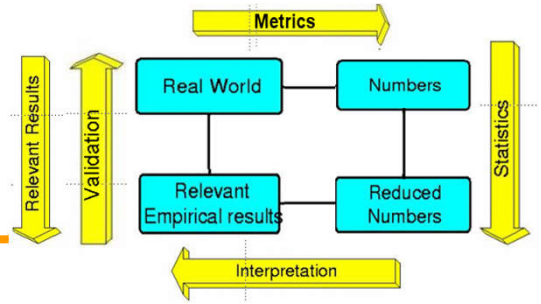
Key Problems



- **How much must we know about an attribute before it is reasonable to consider measuring it?** For example, do we know enough about "size" of Web sites to be able to measure it? Or "complexity" of programs?

- **What meaningful operations can we perform on metrics?** For example, does it make sense to calculate the average productivity for a group of developers, or the average quality of a set of Web sites?

Key Problems (cont.)



- **Using measurement, what meaning statements can we make about an attribute and the entities that posses it?** For example, is it meaningful to talk about doubling a design's quality? If not, how do we compare different designs?
- **How do we know if we have really measured the attribute we wanted to measure?** For example, does the count of the number of "broken links" found on a Web site during integration testing measure the quality of the site? If not, what does the count tell us?

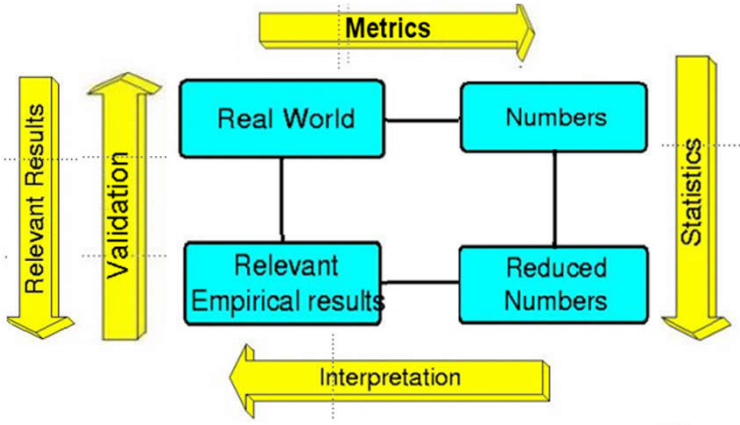
How to Answer These Problems

To answer these questions, we must establish the basics of a theory of measurement

- Measurement Theory
  - Tells us the rules to be applied
- Why Rules?
  - to be consistent in our measurement
  - to provide a basis for interpreting data
- The measurement theory we'll use is the Representational Theory of Measurement

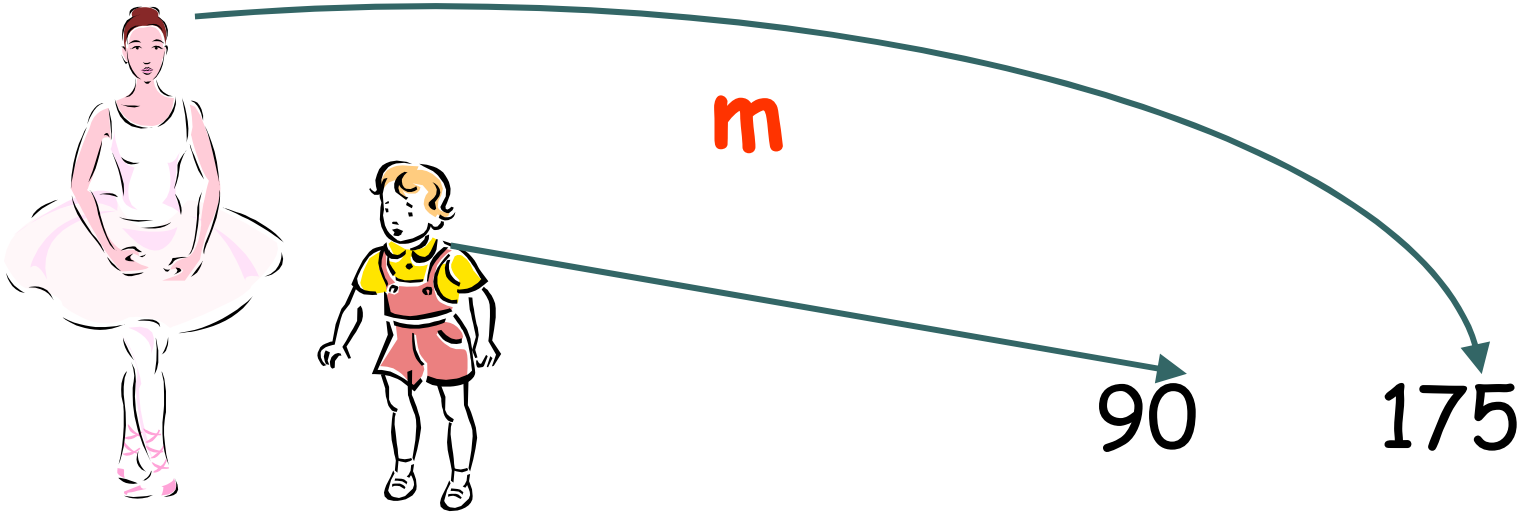
Representation Condition

- Require a metric m
  - Must map the attributes of the entities into numbers
  - Numerical relations of the data preserve empirical relations



Real World

Number System



Mary taller than Tom

$m(\text{Mary}) > m(\text{Tom})$

m satisfy representation condition

x taller than $y \iff m(x) > m(y)$

Example 1 /1

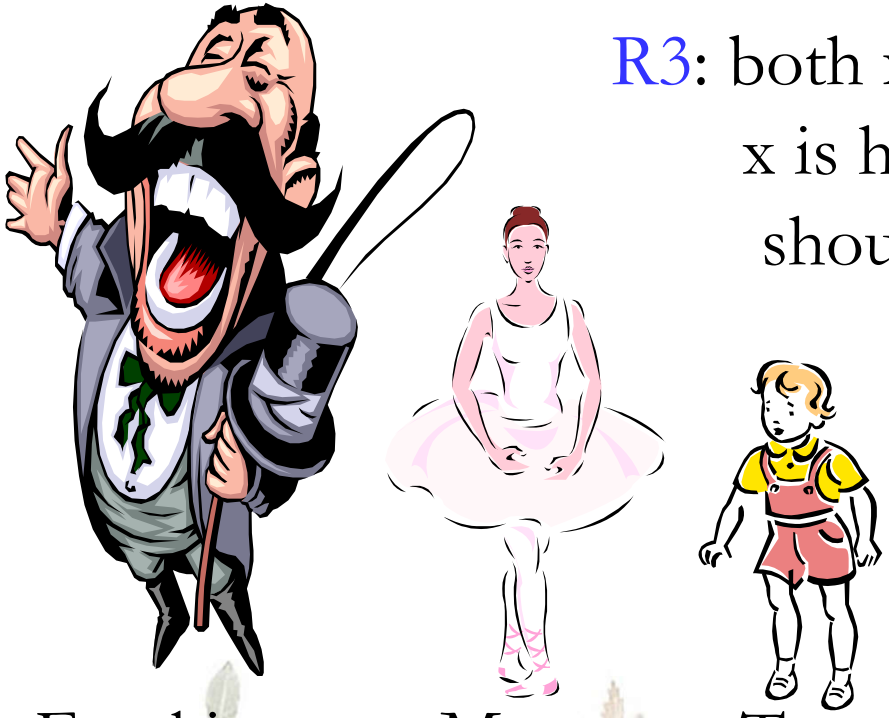
Peoples: Frankie, Mary, Tom

Relations:

R1: *x* is taller than *y* $\Rightarrow$ (x, y) $\in$R1

R2: x is tall $\Rightarrow$ x $\in$R2

R3: both x and y are not higher than z, but x is higher than z if sitting on y's shoulder $\Rightarrow$ (x, y, z) $\in$R3



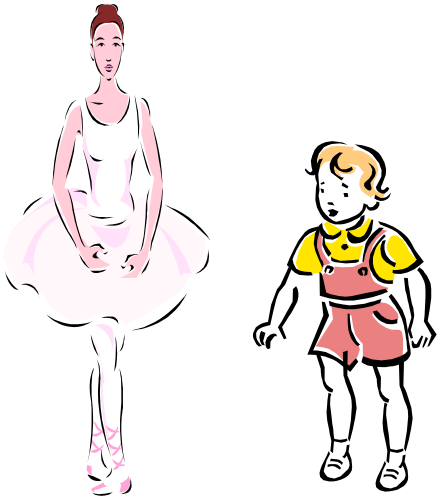
Frankie Mary Tom

Example 1 /2

R1={(Frankie, Mary), (Frankie, Tom), (Mary, Tom)}

Frankie Mary

Frankie Tom



Frankie Tom

Example 1 /3

R2={Frankie, Mary}

R3={(Tom,Mary,Frankie), (Mary,Tom,Frankie)}

Representation Condition

x taller than y $\Leftrightarrow$ $m(x) > m(y)$

x is tall $\Leftrightarrow$ $m(x) > 170$

both x and y are not higher than z, but x is higher than z if sitting on y's shoulder $\Leftrightarrow$

$m(x) < m(z)$

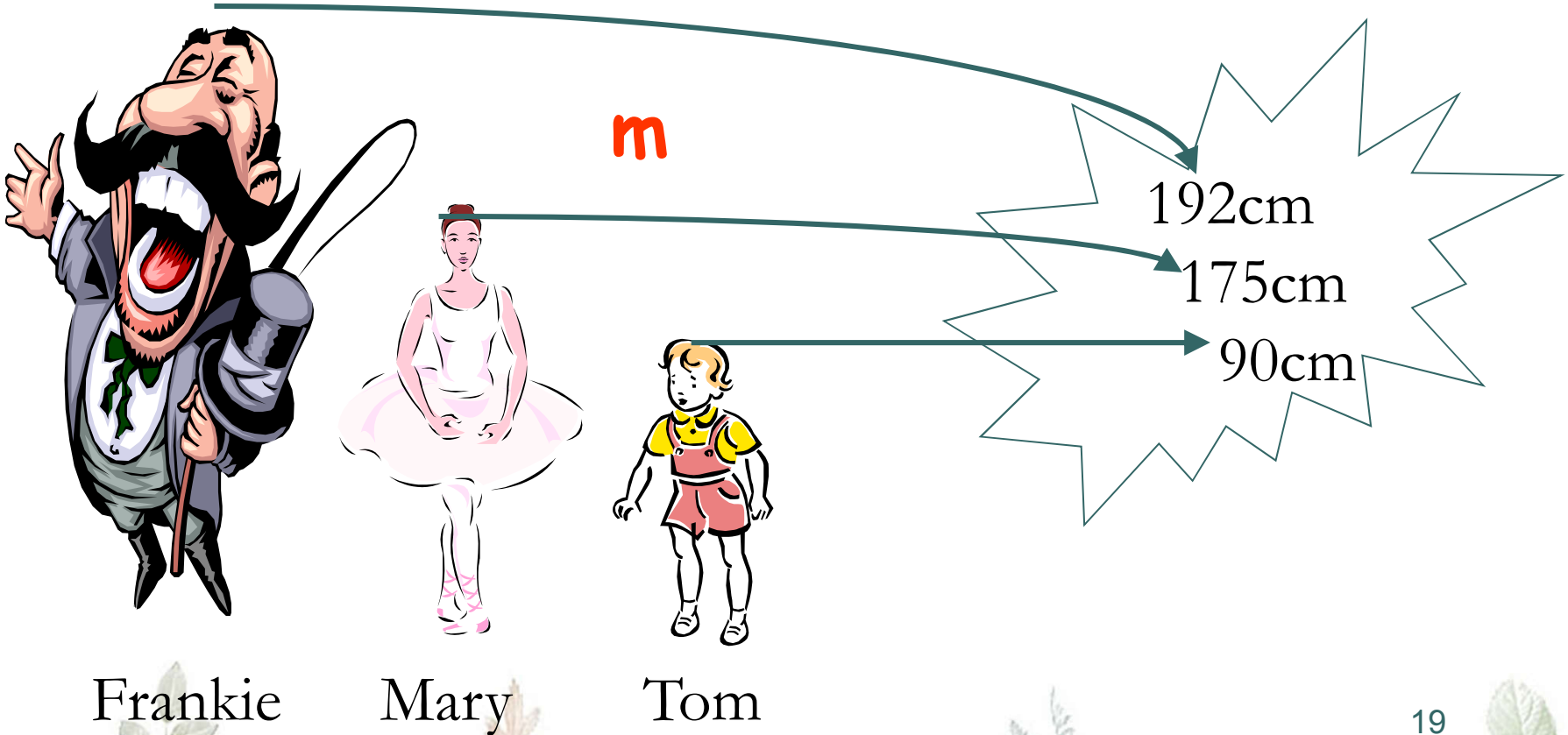
$m(y) < m(z)$

$0.8m(x)+0.7m(y)>m(z)$

Example 1 /4

Measuring

m(Frankie) =192cm
m(Mary) = 175cm
m(Tom) = 90cm



M satisfy representation condition?

- **Entities**: {Frankie, Mary, Tom}
- **Attribute**: Height
- **Empirical Relations**: {R1, R2, R3}

- **Values**: {192cm, 175cm, 90cm}
- **Numerical relation**:

  S1: m(Frankie)>m(Mary), m(Frankie)>m(Tom), m(Mary)>m(Tom)

  S2: m(Frankie)>170cm, m(Mary)>170cm

  S3: 0.8*m(Mary)+0.7*m(Tom)>m(Frankie) 。 。 0.8*m(Tom)+0.7*m(Mary)>m(Frankie)。

Key Stages of Measurement

1) **Identify attribute** for some real world entity
2) **Identify empirical relations** for attribute
3) **Identify numerical relations** corresponding to each empirical relation
4) **Define the model** for the mapping from real world entities to numbers
5) **Check** that numerical relations **preserve** and are preserved by empirical relations

How to Identify Empirical Relations

- The way we perceive the real world

  compare the attributes of entities, not assignment numbers to them

- The arity of empirical relations

  may be unary, binary, or others.

  "is taller than" -----binary

  "is tall" -----unary

  "x is higher than z if sitting on y's shoulder"

  -----ternary

Surveys: Program functionality

	A	B	C	D
A	----	80	10	80
B	20	----	5	50
C	90	95	----	96
D	20	50	4	----

More functionality

(x,y) excess 60%

(C, A), (C,B), (C,D), (A, B), (A, D) **transitive**

Obtain a preliminary understanding of relations by surveys

Example 2 /1

Ranking 4 software products based on user preferences

- Identify attribute: ranking the products (entities) A, B, C and D based on user preferences

- Identify empirical relations: represented by the table

	A	B	C	D
A	-	80	10	80
B	20	-	5	50
C	90	95	-	96
D	20	50	4	-

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

Example 2 /2

	A	B	C	D
A	-	80	10	80
B	20	-	5	50
C	90	95	-	96
D	20	50	4	-

ue greater that 60.

of users prefer A an B.

$p(x) > p(y)$

If $p(x) > p(y)$ and $p(y) > p(z)$

Then $p(x) > p(z)$

Valid pairs (C,A), (C,B), (C,D), (A,B) and (A,D)

- Verification:
  - No conflict between the data collected and the model

Example 2 /3

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

  M(B,C) > 60% then p(B) > p(C)

  M(C,A) > 60% then p(C) > p(A)

  M(A,B) > 60% then p(A) > p(B)

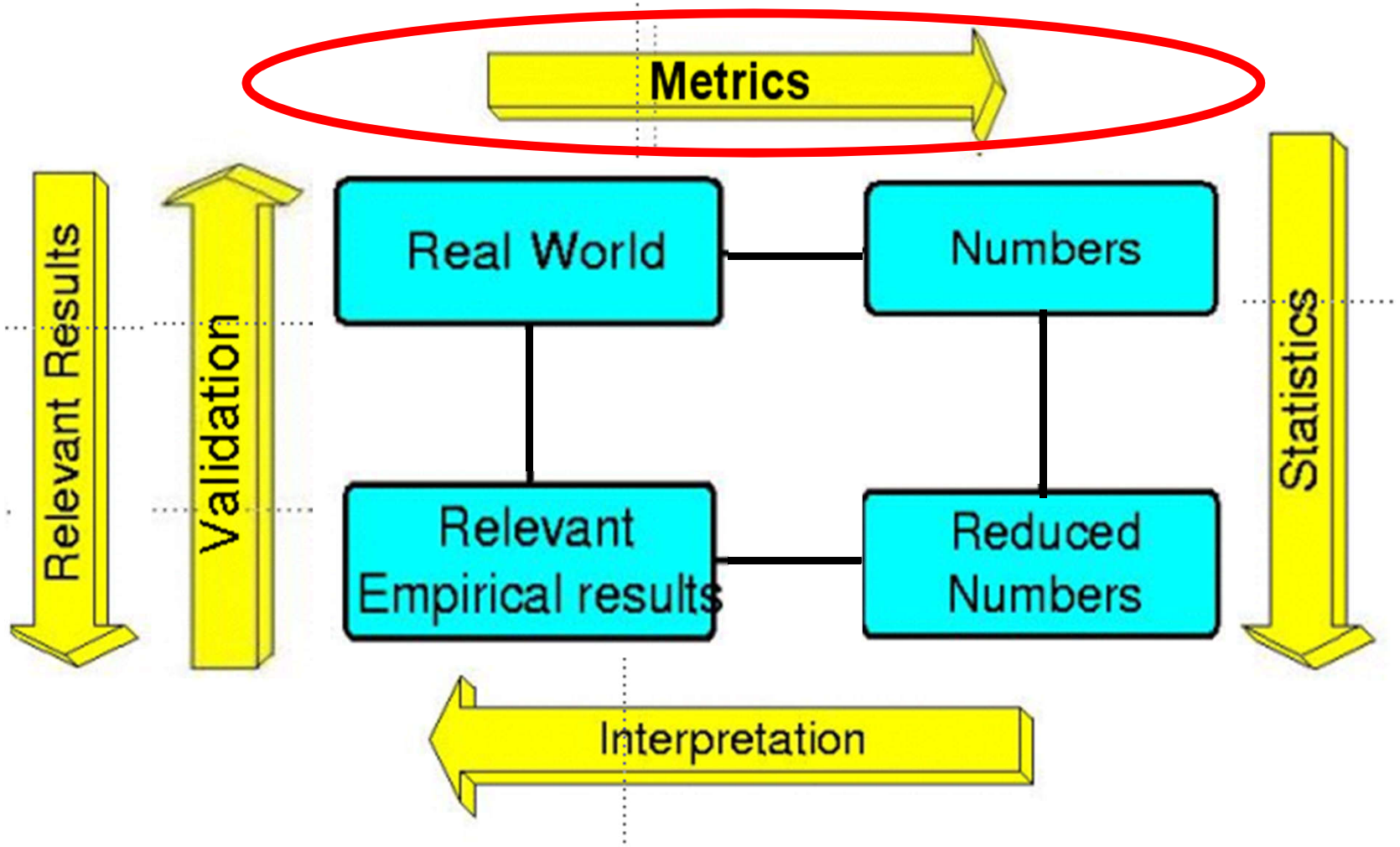
	A	B	C	D
A	-	80	10	80
B	20	-	95	50
C	90	5	-	96
D	20	50	4	-

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

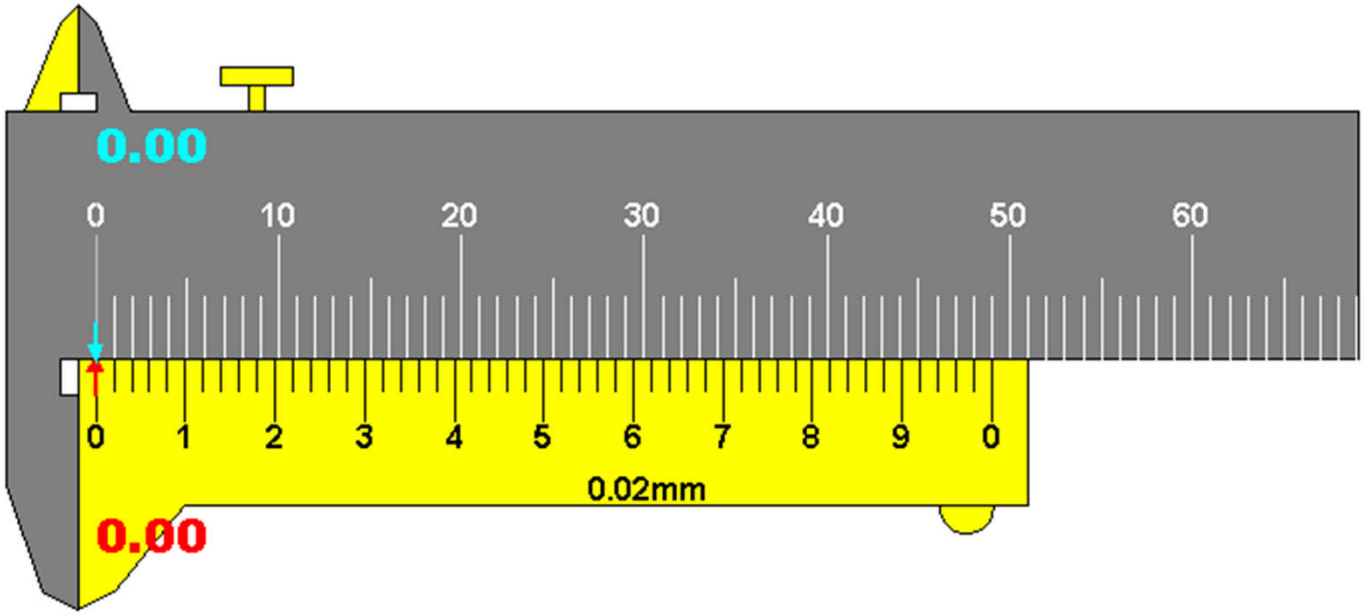
Section 2

Measurement Scales

- Empirical Relational Systems
- Numerical Relational Systems
- Representational Condition
- Scales

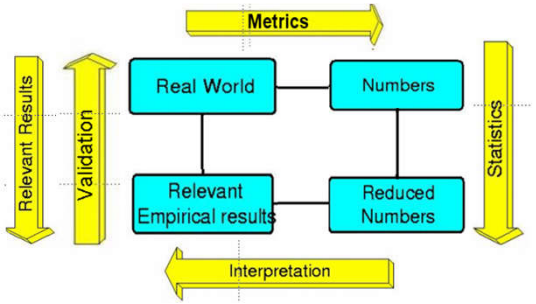

Metrics
Real World
Numbers
Relevant Results
Validation
Statistics
Relevant Empirical results
Reduced Numbers
Interpretation


0 0 1 0 1 1
km/h
MPH
VDO


0.00
0.02mm
0.00

Empirical Relational System

$ERS = \langle E, \mathcal{R}, O \rangle$



Where

- E is a set of entities
- $\mathcal{R}$ is a set of empirical relations R_i

  R_i has an arity n_i, i.e., $R_i \subseteq E^{\text{ni}}$
- O is a set of binary operations among entities

  each binary operation o_j is function

$$o_j : E \times E \rightarrow E$$

Some Examples

- The size of programs

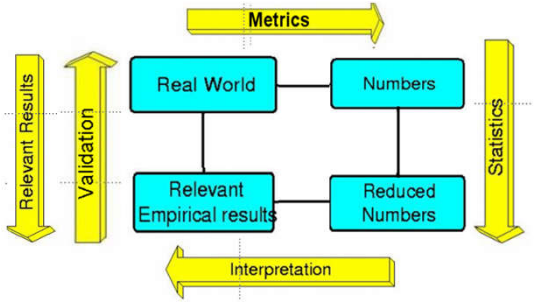
  ERS=<{*program*}, {longer_than}, {⊕}>

- The criticality of software failures

  ERS=<{*software failures*}, {syntactic, semantic, system_crash}>

Numerical Relation Systems

$NRS=<V,\mathfrak{I}, P>$



where

- V is a set of values
- $\mathfrak{I}$ is a set of numerical relations

  S_i has the same arity n_i of the empirical R_i, i.e., $S_i \subseteq E^{ni}$
- P is a set of binary operations among values

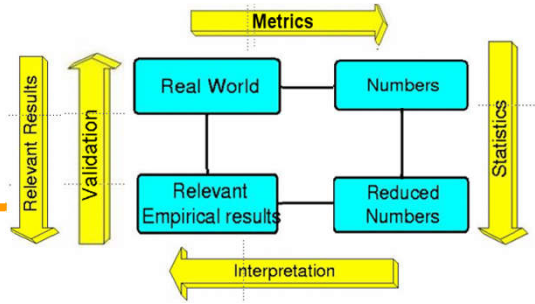
  each binary operation $\bullet_j$ is function

  $$\bullet_j : V \times V \rightarrow V$$

Some Examples

- $NRS=<\{real\}, \{>\}, \{+\}>$
- $NRS=<\{Nature\}, \{>\}, \{-\}>$
- $NRS=<\{all\ strings\}, \{>\}, \{\oplus\}>$

Metric



- **Metric**:

  $m : E \rightarrow V$

  a function that associates a value with each entity

- **Notice**:

  establishes **a link** between the set of entities and the set of values, **regardless of the relations and operations** in the empirical and numerical relational system

Representation Condition

- **Representation condition**

$$\forall i \in 1..n,\ \forall e_1,\ldots,e_n \in E,\ \forall R_i \in \mathfrak{R},\ \forall S_i \in \mathfrak{I}$$
$$R_i(e_1,\ldots,e_n) \Leftrightarrow S_i(m(e_1),\ldots,m(e_n))$$

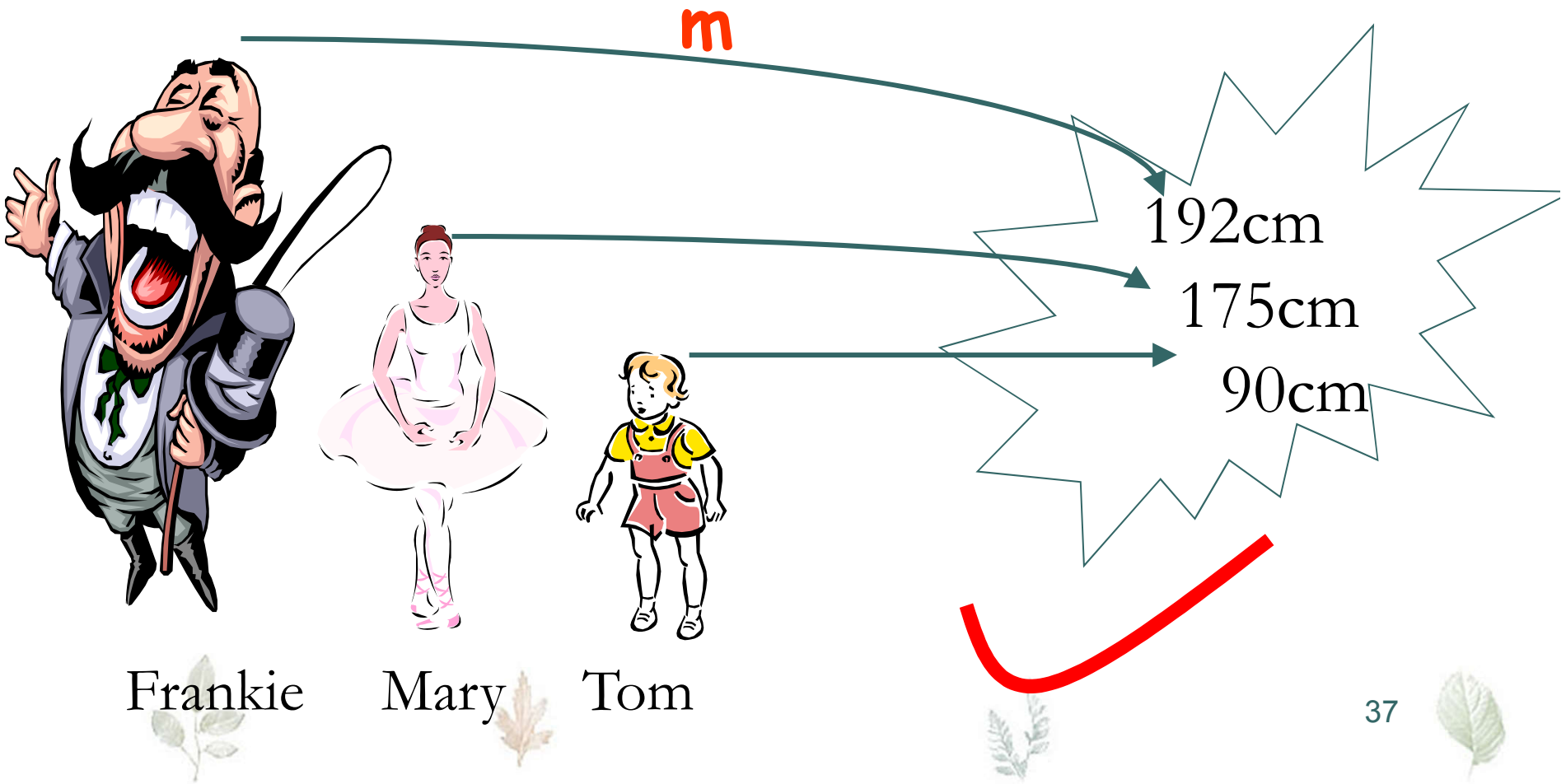
$$\forall j \in 1..m\ \forall e_1, e_2 \in E$$
$$m(e_1\ o_j\ e_2) = m(e_1) \bullet_j m(e_2)$$

Representation Condition (cont.)

Empirical relations	Numerical relations	Representation Condition
taller than	x>y	A is taller than B iff M(A)>M(B)
is tall	x>170	A is tall iff M(A)>170
much taller than	x>y+15	A is much taller than B iff M(A)>M(B)+15
x is higher than y if sitting on z's shoulder	0.8z + 0.7x > y	A is higher than B if sitting C's shoulders iff 0.8M(C)+0.7M(A) >M(B)

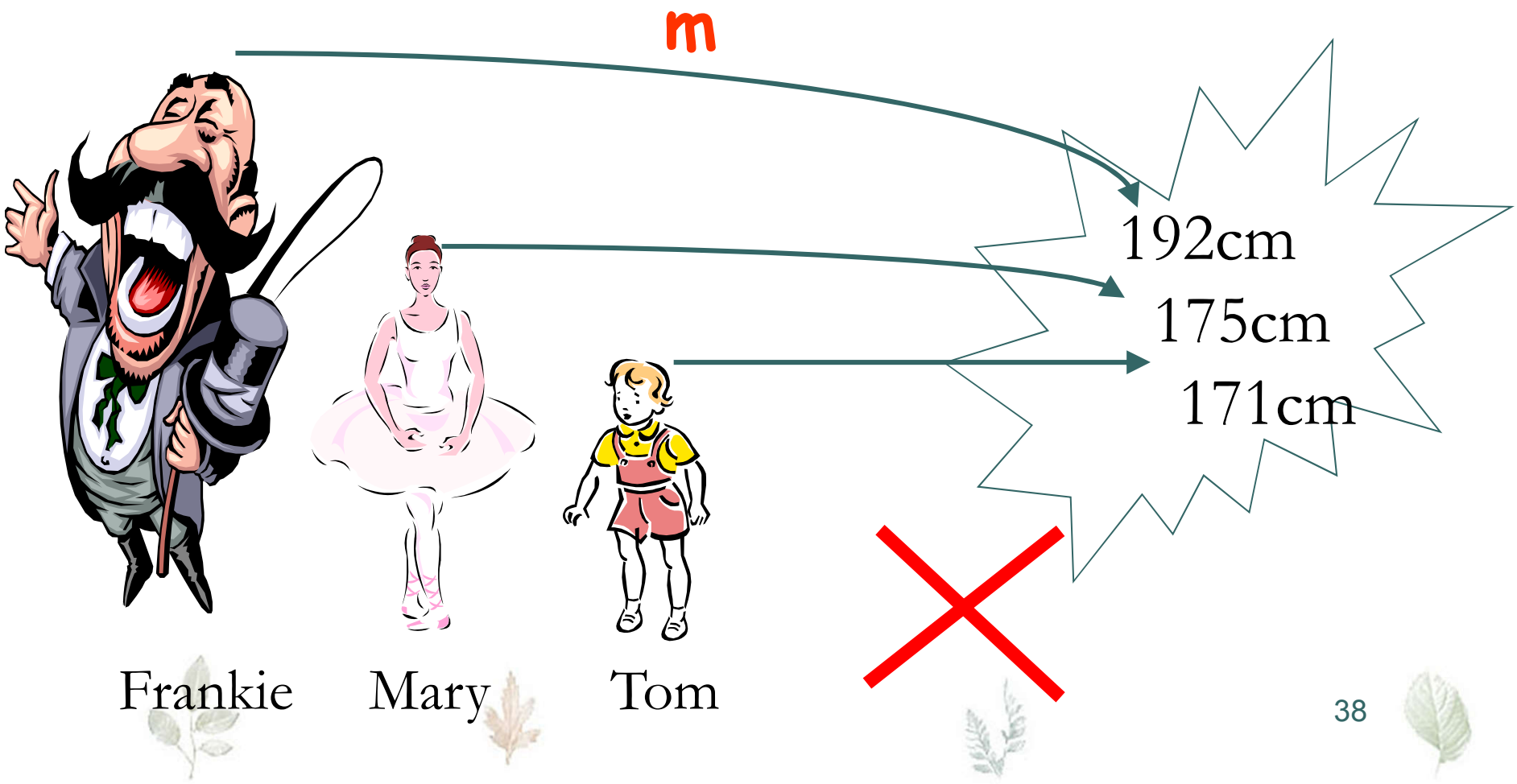
Measuring

m(Frankie) = 192cm
m(Mary) = 175cm
m(Tom) = 90cm

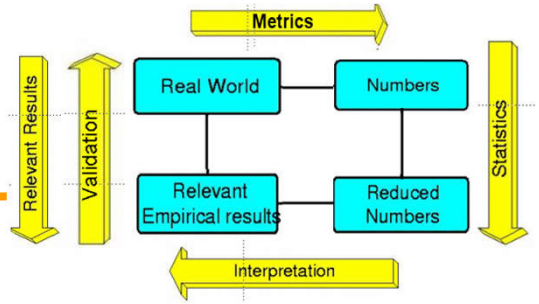


Measuring

m(Frankie) = 192cm
m(Mary) = 175cm
m(Tom) = 171cm



Notice



- There may be many different metrics for a given attribute. Any metric that satisfies the representation condition is a **valid** metric
- The richer the empirical relation system, the fewer the valid metrics

Scales(刻度/尺度/标度)

- A scale is a triple **<ERS, NRS, m>**

where

- *ERS*=<*E*, $\mathcal{R}$, *O*> is an empirical relational sys
- *NRS*=<*V*,$\Im$, *P*> is a numerical relational sys
- m : E→V is a metric that **satisfies the Representation condition**

Sometime, we refer to m alone as the scale

Related Problems

- **How** do we determine one numerical relation system in preference to another?
- **How** do we know whether a particular empirical relation system has a representation in a given numerical relation system? (**existence**)
- **What** do we do when we have a number of different possible representations (and hence many scales) in the same numerical relation system? (**Uniqueness**)

Related Problems (cont.)

- Answer to 1: Use the real number as possible
- Answer to 2: **Representation theorem**
  - D.H. Krantz. Foundations of Measurement.1971
  - F.S. Roberts. Measure Theory with Applications to Decision Making Utility and the Social Sciences. 1979
- Answer to 3: **Transformation**
  - Height of persons can be measured with different scales, e.g., meters, centimeters
  - If m is a distance metric, then $m'=\alpha m$ is also a distance metric.However, $m'=m^2$ is not

Related Problems (cont.)

- **Admissible Transformation**

  Given a scale <*ERS*, *NRS*, *m*>, a transformation *f* is admissible if <*ERS*, *NRS*, *m*′ > is a scale, where *m*′ = *f* ∘ *m* is the composition of *f* and *m*

- **Rescaling (換算)**

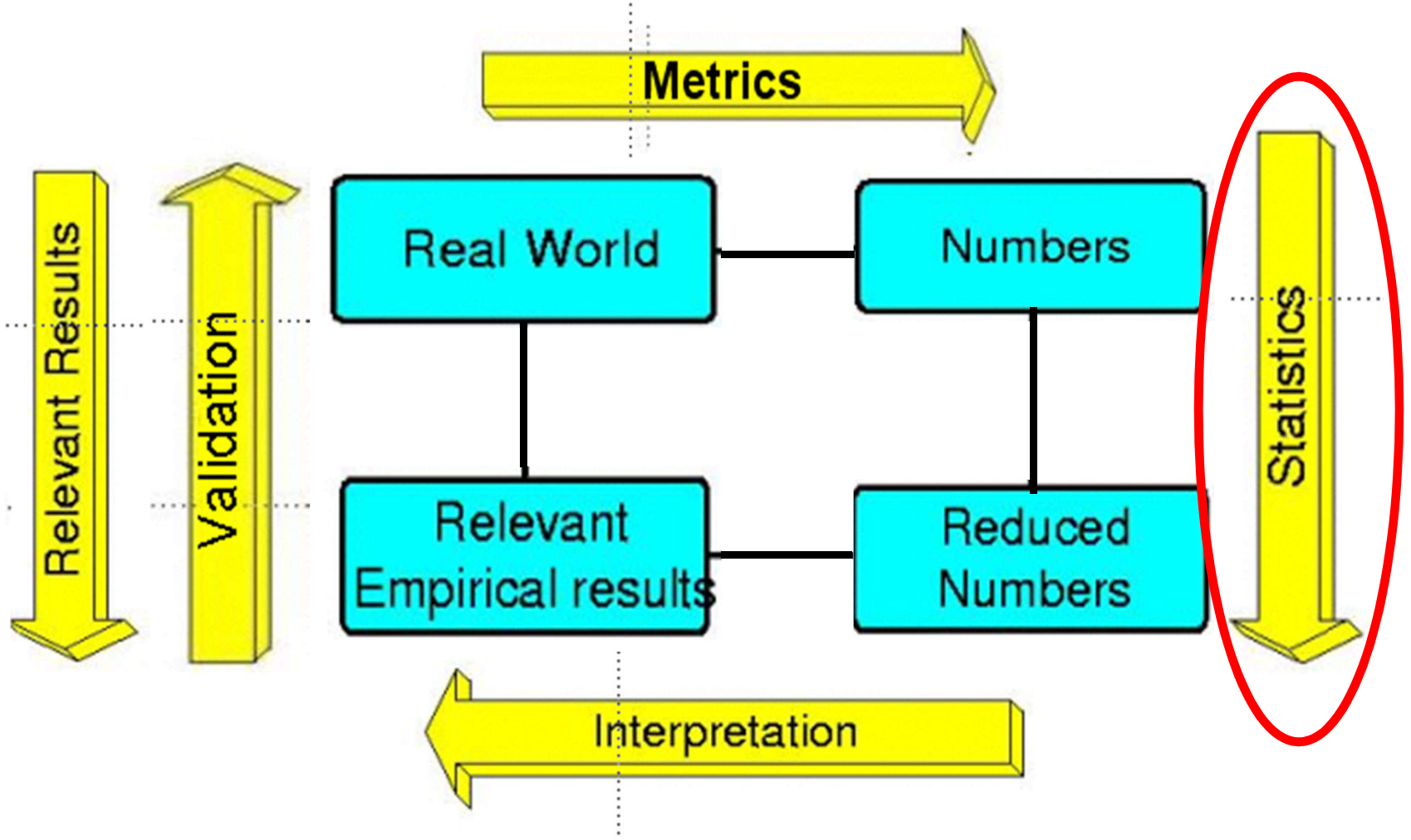
  m′ is called as a rescaling of *m*

different types of scales permit different types of admissible transformation

Section 3

Scale Types

- **Nominal Scale**
- **Ordinal Scale**
- **Interval Scale**
- **Ratio Scale**
- **Absolute Scale**


Metrics
Real World
Numbers
Relevant Results
Validation
Statistics
Relevant
Empirical results
Reduced
Numbers
Interpretation

Nominal Scale

<ERS, NRS, m>

Definition:A scale is a nominal one if it divides the set of entities into categories, with no particular ordering among them

- The ERS 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 metric, but there is no notion of magnitude associated with the numbers or symbols

- Admissible transformation

  All one-to-one transformation

- Notice

  It makes no sense to carry out any arithmetic operations on these numbers or order them

- Example: the location of faults

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

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

Ordinal Scale

<ERS, NRS, m>

Definition: A scale is an ordinal one if it divides the set of entities into categories that are ordered

- The ERS 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

- Admissible transformation

  All strictly monotonic functions

- Notice

  Although we can compare values, we still cannot use them for arithmetic operations

- Example: complexity of modules

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

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

Interval Scale

<ERS, NRS, m>

Definition: In an interval scale, the distance between two values is the basis for meaningful statements

- An interval scale preserves order, as with an ordinal scale
- An interval scale preserves differences but not ratios
- Addition and subtraction are acceptable on the interval scale, but not multiplication and division

- Admissible transformation

  $m' = am+b \quad (a > 0)$ (a subset of the monotonic functions that are admissible for ordinal scales.)

- Example

$$M_1(x) = \begin{cases} 15c & \text{Shanghai} \\ 20c & \text{Beijing} \\ 25c & \text{Zhengzhou} \\ 30c & \text{Nanjing} \\ 35c & \text{Wuhan} \end{cases} \qquad M_2(x) = \begin{cases} 59F & \text{Shanghai} \\ 68F & \text{Beijing} \\ 77F & \text{Zhengzhou} \\ 86F & \text{Nanjing} \\ 95F & \text{Wuhan} \end{cases}$$

Ratio Scale

<ERS, NRS, m>

Definition:A ratio scale allows the definition of meaningful statements of the kind “twice as much” on the single values of the metric.

- 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, know as units
- All arithmetic can be meaningfully applied to it

- Admissible transformation:

  $m' = am \quad (a>0)$

- Notice:

  It makes sense to add and subtract ratio metrics, and to multiply and divide them by constants

- Example:

  The length of physical objects

  The length of software code




Absolute Scale

<ERS, NRS, m>

Definition: A scale that only allows for the identify transformation.

- The measurement for an absolute scale is made simply by counting the number of elements in the entity set
- There is only one possible measurement mapping, namely actual number
- All arithmetic analysis of the resulting count is meaningful

- Admissible transformation:

  the identity transformation: $m' = m$

- Example:
  - LOC is a ratio scale of program length
  - LOC is a absolute scale of the attribute “number of lines of code” of a program
  - “number of years” is a ratio-scale metric of a person’s age



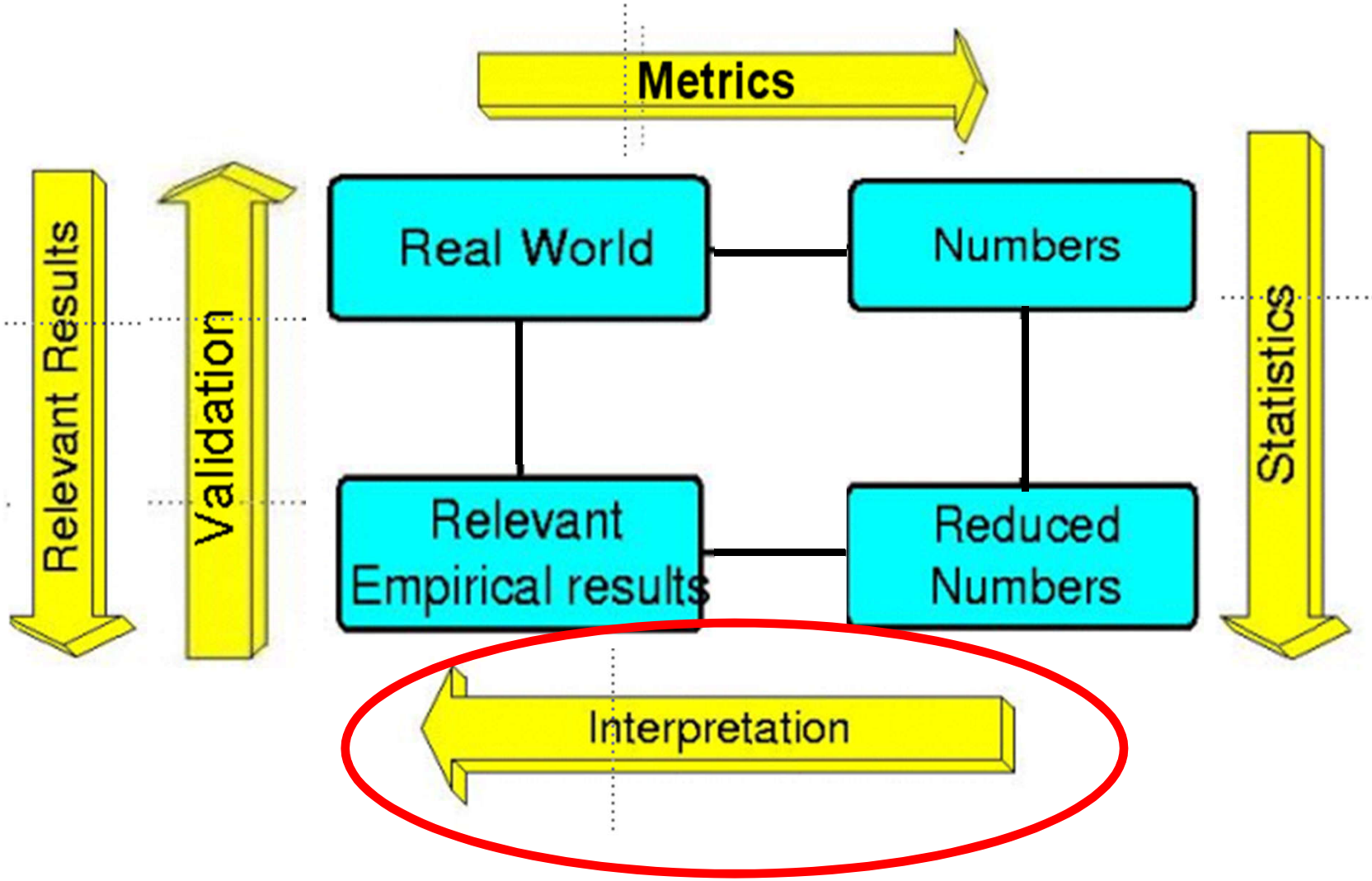

Summary: scales of measurement

Scale Types	Admissible Transformation	Examples
Nominal	Bijections	Labeling, classifying entities
Ordinal	Monotonically increasing	Preference, hardness, air quality, intelligence tests
Interval	Positive linear	Relative time, temperature
Ratio	Proportionality	Time interval, length
Absolute	Identity	Counting entities

Section 4

Meaningfulness in Measurement

- The notion of meaningfulness
- Operations on metrics


Metrics
Real World
Numbers
Relevant Results
Validation
Statistics
Relevant Empirical results
Reduced Numbers
Interpretation

Discussion

The President of the United States is 125 years old

true or false?

meaningful?

The notion of meaningfulness

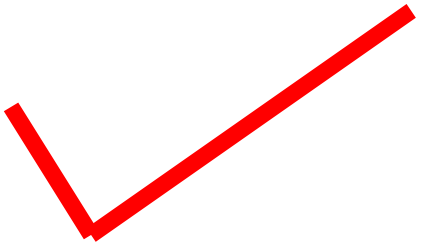
- Definition

A statement involving measurement is meaningful iff its **truth value** is **invariant of transformations of allowable scales**

statements and inferences based on data from measurement scales are meaningful if and only if their truth (or falsity) remains unchanged under all admissible transformations of all the scales involved

Examples

1) Fred is twice as tall as Jane



tallness is **a ratio scale: M'=aM**

If M and M' are different metrics of height, then

M(Fred) =2M(Jane) and M'(Fred)=2M'(Jane)

are **either both true or both false**

Fred is 180cm tall
Jane is 90cm tall
} M(Fred)=2*M(Jane)

same as:

Fred is 1.80m tall
Jane is 0.9m tall
} M'(Fred)=2*M'(Jane)

1m = 100cm

Examples (cont.)

2) The temperature in Tokyo is twice that in London

C(Tokyo) = 40 °
C(London) = 20 °

C(Tokyo)=2*C(London)

Temperature is an interval scale

F = 9/5 C+32

F(Tokyo)=104 °
F(London)= 68 °

C(Tokyo) ≠2*C(London)

Examples (Cont.)

3) The difference in temperature between Tokyo and London today is twice what it was yesterday

Yesterday

C(Tokyo) = 35 °
C(London) = 25 °
} 10

Today

C(Tokyo) = 40 °
C(London) = 20 °
} 20

|C(TokyoT)-C(LondonT)|=2*(C(TokyoY)-C(LondonY)

F(Tokyo) = 95 °
F(London) = 77 °
} 18

F(Tokyo) = 104 °
F(London) = 68 °
} 36

|F(TokyoT)-F(LondonT)|=2*(F(TokyoY)-F(LondonY)

Examples (Cont.)

4) The President of the United States is 125 years old

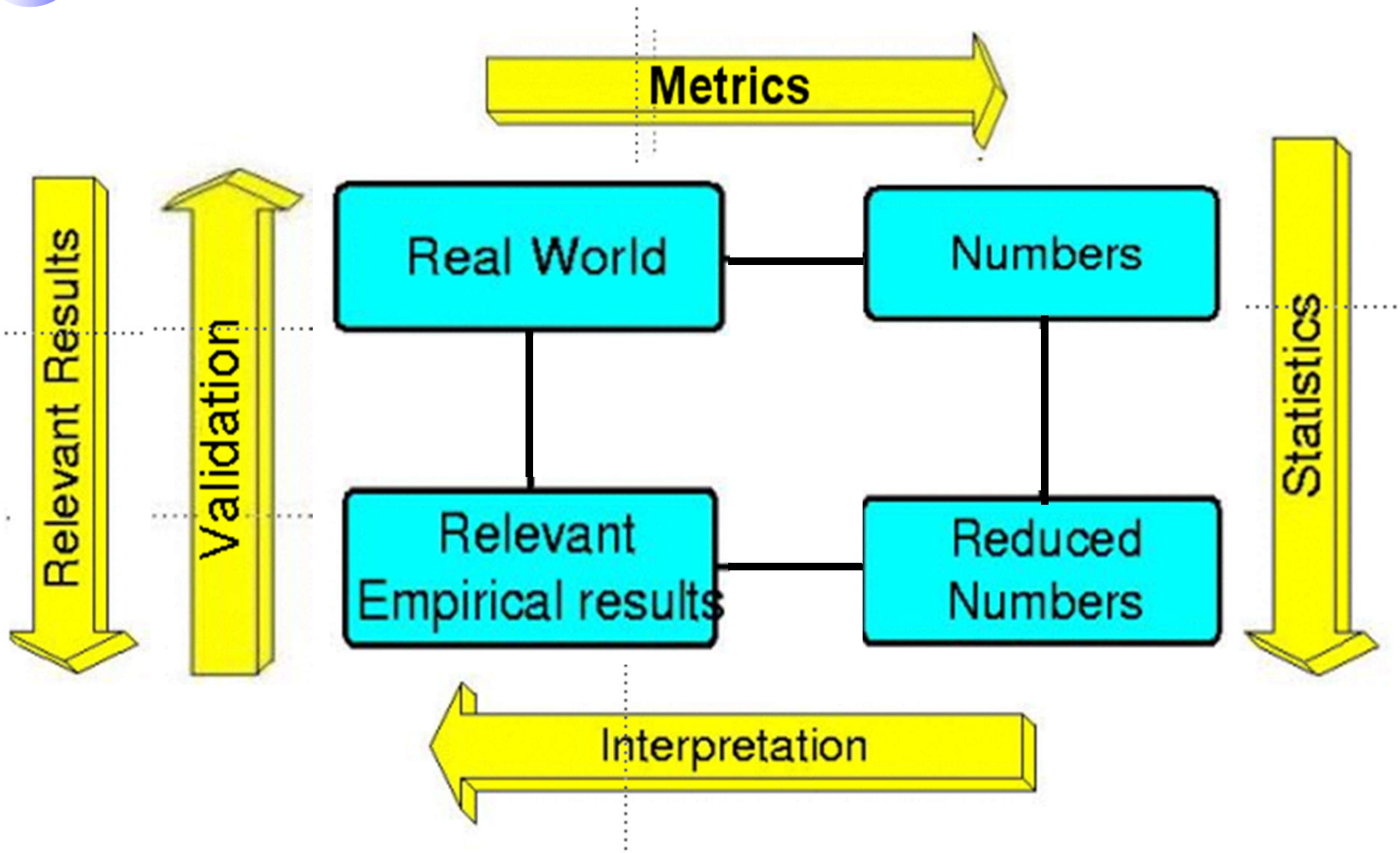
5) A semantic error takes twice as long to fix as a syntactic error

6) A semantic error is twice as complex as a syntactic error

Scale types and Statistics(P59)

Scale Type	**Defining Relations**	**Examples of statistic**	**Appropirate statistical tests**
Nominal	Equivalence	Mode Frequency	Non-parametric
Ordinal	Equivalence Greater than	Median Percentile Spearman r Kendall W	Non-parame tric
Interval	Equivalence Greater than Known ratio of any intervals	Mean Standard deviation Person product-moment correlation Multiple product-moment correlation	Non-para metric
Ratio	Equivalence Greater than Known ratio of any intervale Known ratio of any two scale values	Geometric mean Coefficient of variation	Non-parametric and parametric

Summary

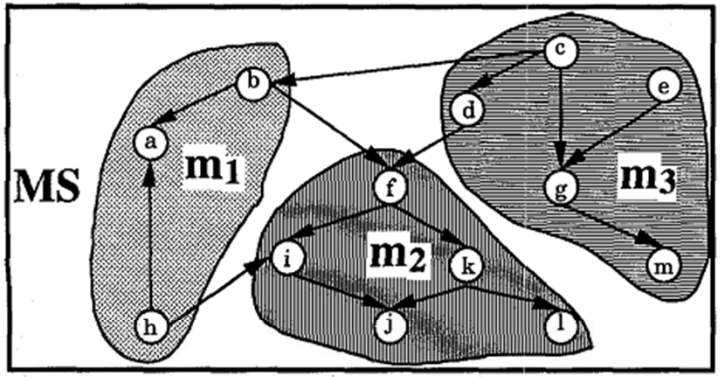

Metrics
Real World
Numbers
Relevant Results
Validation
Statistics
Relevant Empirical results
Reduced Numbers
Interpretation

实例分析

- L. Briand, et al. Property-based software engineering measurement. IEEE TSE 1996
- J. Al-Dallal, L. Briand. A precise method-method interaction-based cohesion metric for object-oriented classes. ACM TOSEM 2012

L. Briand, et al. Property-based software engineering measurement. IEEE TSE 1996

DEFINITION 3: Size. *The size of a system S is a function* Size(S) *that is characterized by the following properties* Size.1-Size.3.



PROPERTY SIZE.1: Nonnegativity. The size of a system S = <E, R> is nonnegative

$Size(S) \geq 0$ (Size.I) □

PROPERTY SIZE.2: Null Value. The size of a system S = <E, R> is null if E is empty

$E = \varnothing \Rightarrow Size(S) = 0$ (Size.II) □

PROPERTY SIZE.3: Module Additivity. The size of a system S = <E, R> is equal to the sum of the sizes of two of its modules $m_1 = \langle E_{m1}, R_{m1} \rangle$ and $m_2 = \langle E_{m2}, R_{m2} \rangle$ such that any element of S is an element of either m_1 or m_2

$(m_1 \subseteq S \text{ and } m_2 \subseteq S \text{ and } E = E_{m1} \cup E_{m2} \text{ and } E_{m1} \cap E_{m2} = \varnothing)$
$\Rightarrow Size(S) = Size(m_1) + Size(m_2)$ (Size.III) □

L. Briand, et al. Property-based software engineering measurement. IEEE TSE 1996

思考1：LOC是size度量吗？

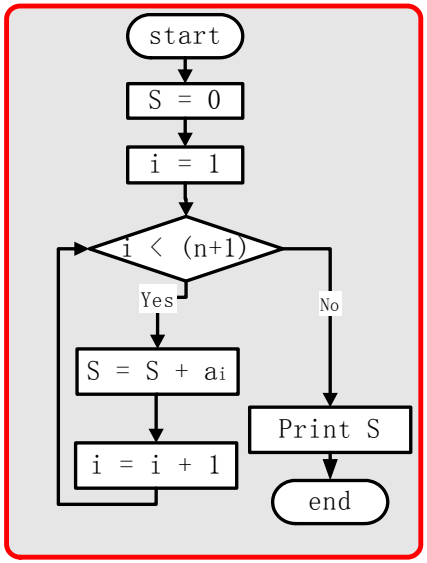
思考2：WMC是size度量吗？

圈复杂性 $v(G) = e - n + 2p$

$= 8 - 8 + 2 \times 1$

$= 2$

```
void sum()
{
    S=0;
    i=1;
    while(i<n+1)
    {
      S=S+a[i];
      i=i+1;
    }
  printf("S is %d", s);
}
```



L. Briand, et al. Property-based software engineering measurement. IEEE TSE 1996

思考3：Size的3个性质描述的是$\mathfrak{R}$还是$\mathfrak{I}$？

- A scale is a triple <ERS, NRS, m>， where
  - *ERS*=<*E*, $\mathfrak{R}$, *O*> is an empirical relational sys
  - *NRS*=<*V*,$\mathfrak{I}$, *P*> is a numerical relational sys
  - m : E→V is a metric that **satisfies the Representation condition**

$$(m_1 \subseteq S \text{ and } m_2 \subseteq S \text{ and } E = E_{m1} \cup E_{m2} \text{ and } E_{m1} \cap E_{m2} = \varnothing) \Rightarrow Size(S) = Size(m_1) + Size(m_2)$$

$$(m_1 \subseteq S \text{ and } m_2 \subseteq S \text{ and } E = E_{m1} \cup E_{m2} \text{ and } E_{m1} \cap E_{m2} = \varnothing) \Rightarrow S \approx m_1 \oplus m_2)$$

L. Briand, et al. Property-based software engineering measurement. IEEE TSE 1996

思考4：如果一个度量满足Size的3个性质，它是一个“好”的度量吗？

Size: 3个性质

Length: 5个性质

Complexity: 5个性质

Cohesion: 4个性质

Coupling: 5个性质

J. Al-Dallal, L. Briand. A precise method-method interaction-based cohesion metric for object-oriented classes. ACM TOSEM 2012

提出了一个新的内聚性度量LSCC

证明它满足Cohesion的4个性质

5. THEORETICAL VALIDATION

We validate LSCC using the properties for a class cohesion metric proposed by Briand et al. [1998] and discussed in Section 2.1.

PROPERTY LSCC.1. *The LSCC metric complies with the non-negativity and normalization property.*

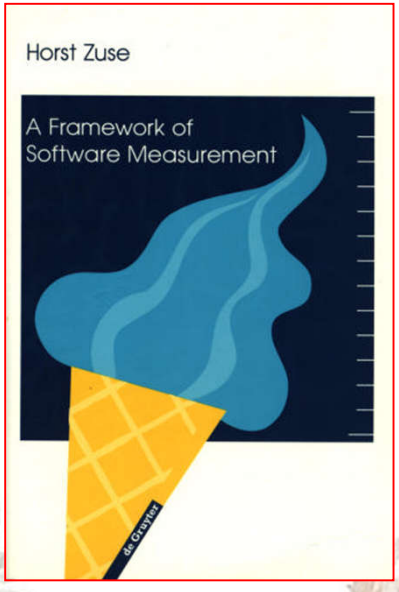
PROOF. The minimum value for the LSCC metric for a class is 0 when the class has either (1) several methods and no attributes or (2) several methods such that none of their pairs share a common attribute. The maximum value for the LSCC metric for a class is 1 when the class has (1) one or more attributes and no methods, (2) one method

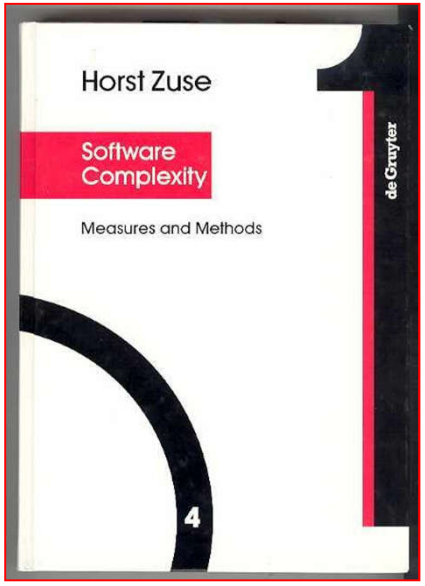
Summary

- Scale?
- Scale type?
- Meaningful?
- Operations on metrics?

Further reading

- S. Morasca. Foundations of a weak measurement-theoretic approach to software measurement. 2003
- A. Meneely, et al. Validating software metrics: a spectrum of philosophies. ACM TOSEM 2012





47个标准


Validity
#24: Internal Validity (4)
#17: External Validity (4)
#10: Construct Validity (1)
#16: Empirical Validity (2)
#42: Theoretical Validity (2)
#42: Theoretical Validity (2)
#16: Empirical Validity (2)
#20: Instrument Validity (1)
#35: Protocol Validity (1)
#41: Stability (3)
#47: Usability (1)
#11: Constructiveness (1)
#45: Underlying Theory Validity (1)
#6: Attribute Validity (1)
#1: A Priori Validity (4)
#5: Association (2)
#7: Causal Model Validity (1)
#13: Discriminative Power (1)
#19: Improvement Validity (1)
#36: Rank Consistency (1)
#38: Repeatability (1)
#43: Traceability (1)
#2: Actionability (3)
#15: Economic Productivity (1)
#23: Internal Consistency (1)
#26: Metric Reliability (1)
#28: Non-exploitability (1)
#44: Transformation Invariance (1)
#9: Content Validity (1)
#34: Product and Process Relevance (3)
#12: Definition Validity (6)
#30: Notation Validity (1)
#45: Underlying Theory Validity (5)
#18: Factor Independence (1)
#8: Causal Relationship Validity (3)
#32: Predictability (5)
#27: Non-collinearity (1)
#33: Prediction System Validity (2)
#39: Representation Condition (6)
#4: Appropriate Granularity (2)
#21: Increasing Growth Validity (1)
#22: Interaction Sensitivity (1)
#25: Monotonicity (1)
#29: Non-uniformity (1)
#31: Permutation Validity (1)
#37: Renaming Insensitivity (1)
#3: Appropriate Continuity (1)
#14: Dimensional Consistency (2)
#40: Scale Validity (4)
#46: Unit Validity (1)
Legend
"is a" Relationship
(#ref number) Criterion Name (found in X sources)
(#ref number) Idea or Group (found in X sources)

Thanks for your time and attention!

练习：用Understand收集数据

(1) Understand介绍

(2) 下载gcc 2.8

http://ftp.gnu.org/gnu/gcc/gcc-2.8.0.tar.gz

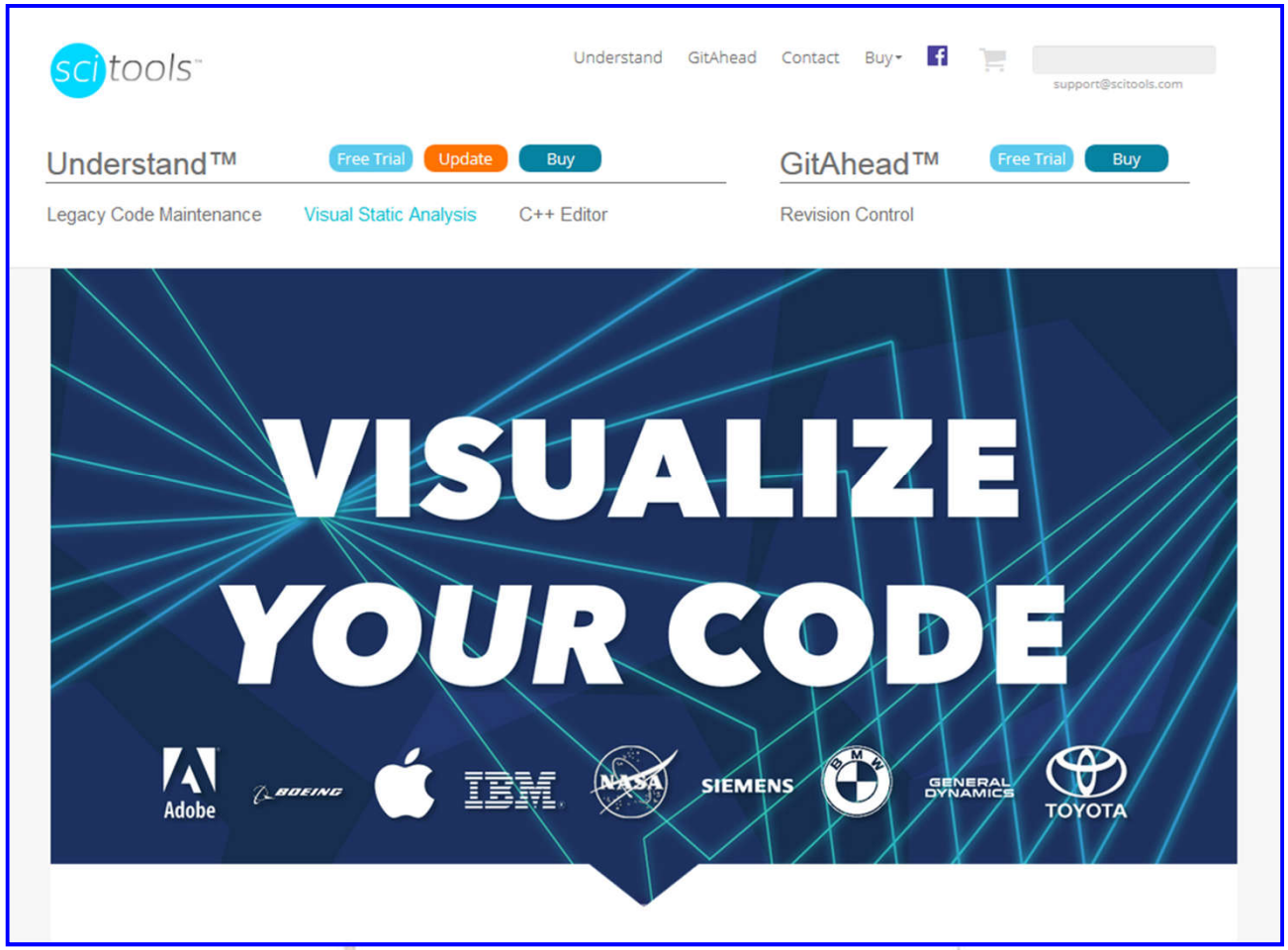
(3) 编写一个Perl脚本，为gcc 2.8上的函数收集，以.csv文件格式输出：

函数名, 相对路径, CountLineCode, CountPath, Cyclomatic, MaxNesting, Knots, CountInput, CountOutput

在10月29日前提交：Perl脚本以及数据集

Understand

http://www.scitools.com/



All Industries
Aerospace
Automotive
Defense
Electronics
Energy
Entertainment
Financial
Gaming
Healthcare
Manufacturing
Retail
Software
Technology
Telecommunications
Transportation

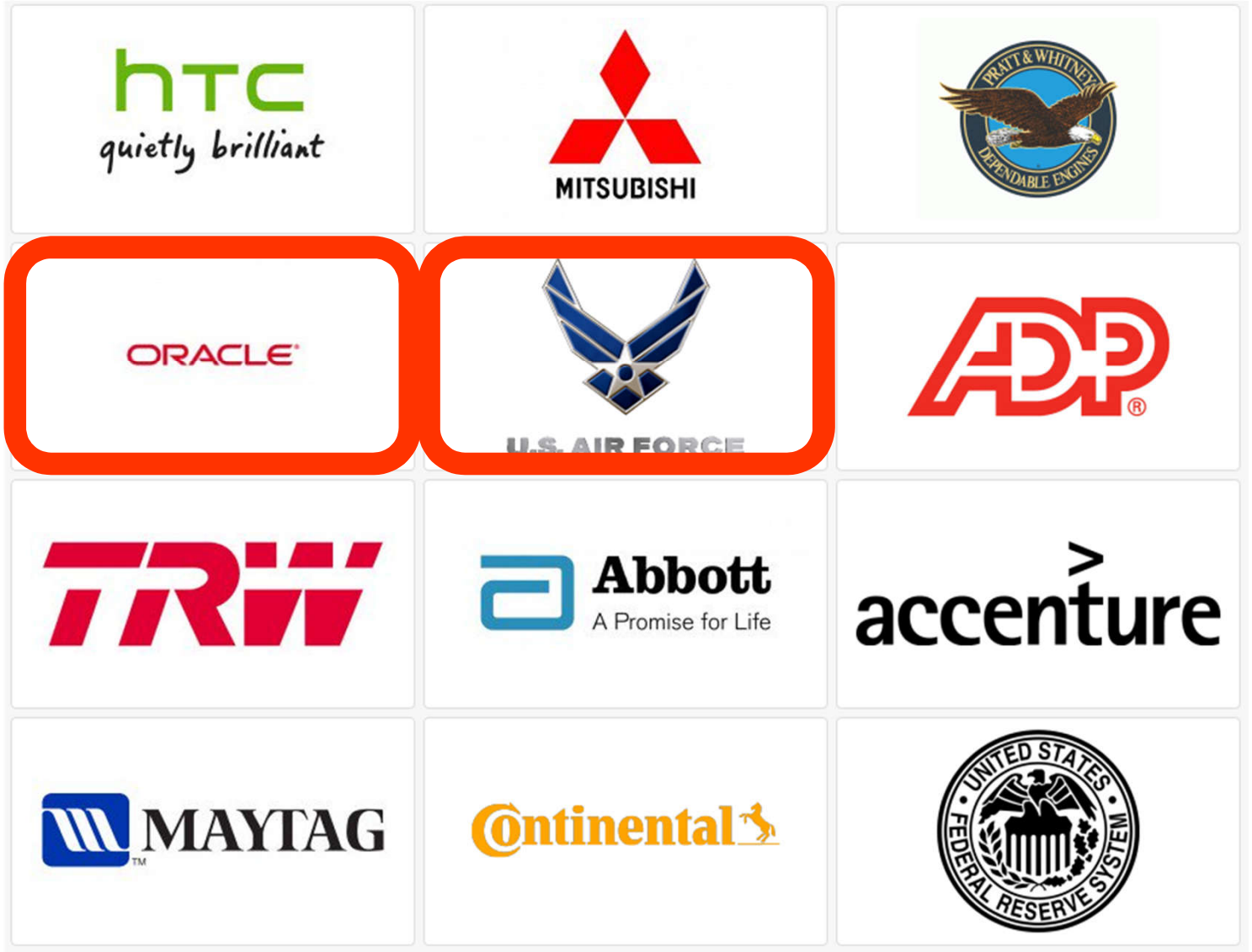

AleniaAermacchi
A Finmeccanica Company
Honeywell
JOHN DEERE
QUALCOMM
United Technologies
IDEXX
LABORATORIES
BMW
Disney
INTER
ACTIVE
UNITED STATES
MARINE CORPS
THE HOME DEPOT


Western Digital
LG
intuit
ConocoPhillips
maxis
IBM
THE AEROSPACE CORPORATION
U.S.ARMY
NOKIA
Mentor Graphics
RAIN BIRD


CardinalHealth
L3 communications
CISCO
Ford
Go Further
SONY
ExxonMobil
LOCKHEED MARTIN
Canon
3M
Rockwell Collins
Donaldson
Filtration Solutions
Novell


MOTOROLA
Aurora
FLIGHT SCIENCES
Johnson & Johnson
Pitney Bowes
NASA
COMCAST
AMD
PHILIPS
CATERPILLAR®
DUPONT
Google™
VOLVO

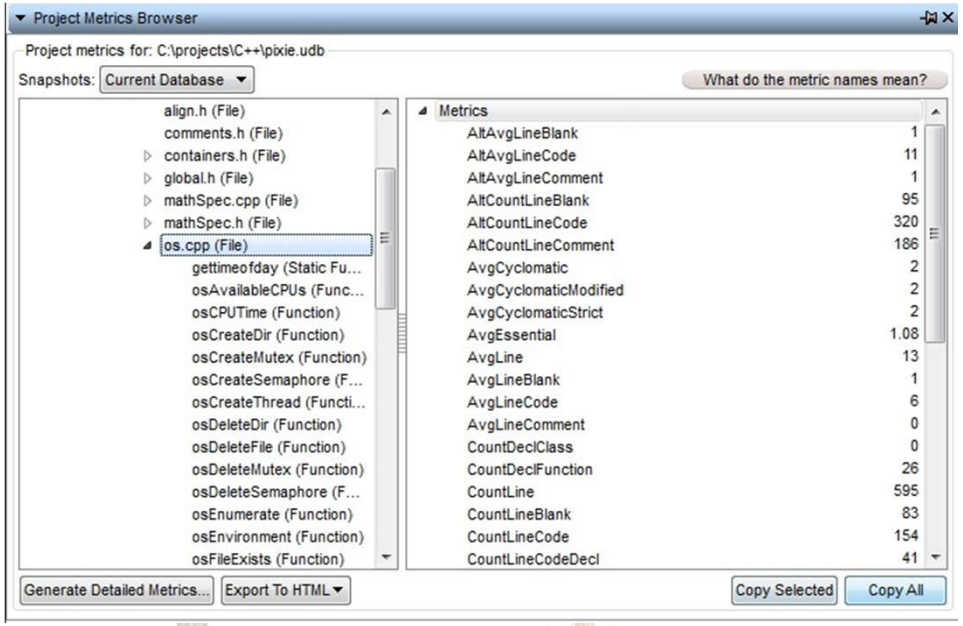

memorex
SIEMENS
2K GAMES
TOYOTA
Walt Disney Imagineering
BROADCOM
Adobe
SAMSUNG
CUBIC
Audi
Alcatel·Lucent
MINISTRY OF DEFENCE
UNITED KINGDOM

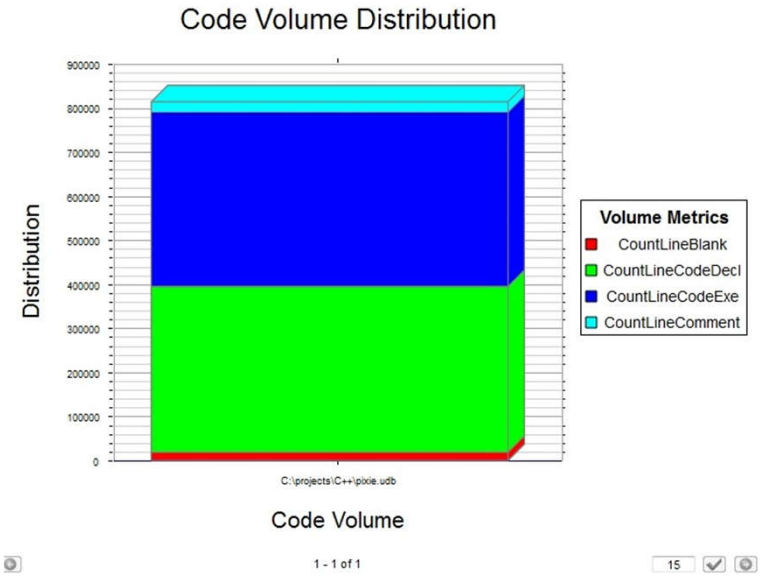

htc
quietly brilliant
MITSUBISHI
PRATT & WHITNEY
DEPENDABLE ENGINES
ORACLE
U.S. AIR FORCE
ADP
TRW
Abbott
A Promise for Life
accenture
MAYTAG
Continental
UNITED STATES
FEDERAL RESERVE SYSTEM


DELL
Raytheon
SWISS POST
DEPARTMENT OF JUSTICE
FEDERAL BUREAU OF INVESTIGATION
Chevron
hp
HYUNDAI
citi
BOEING

Understand: Metrics

1. Standard metrics
   - ① Complexity metrics (20+)
   - ② Volume metrics (40+)
   - ③ Object-oriented metrics (~30)

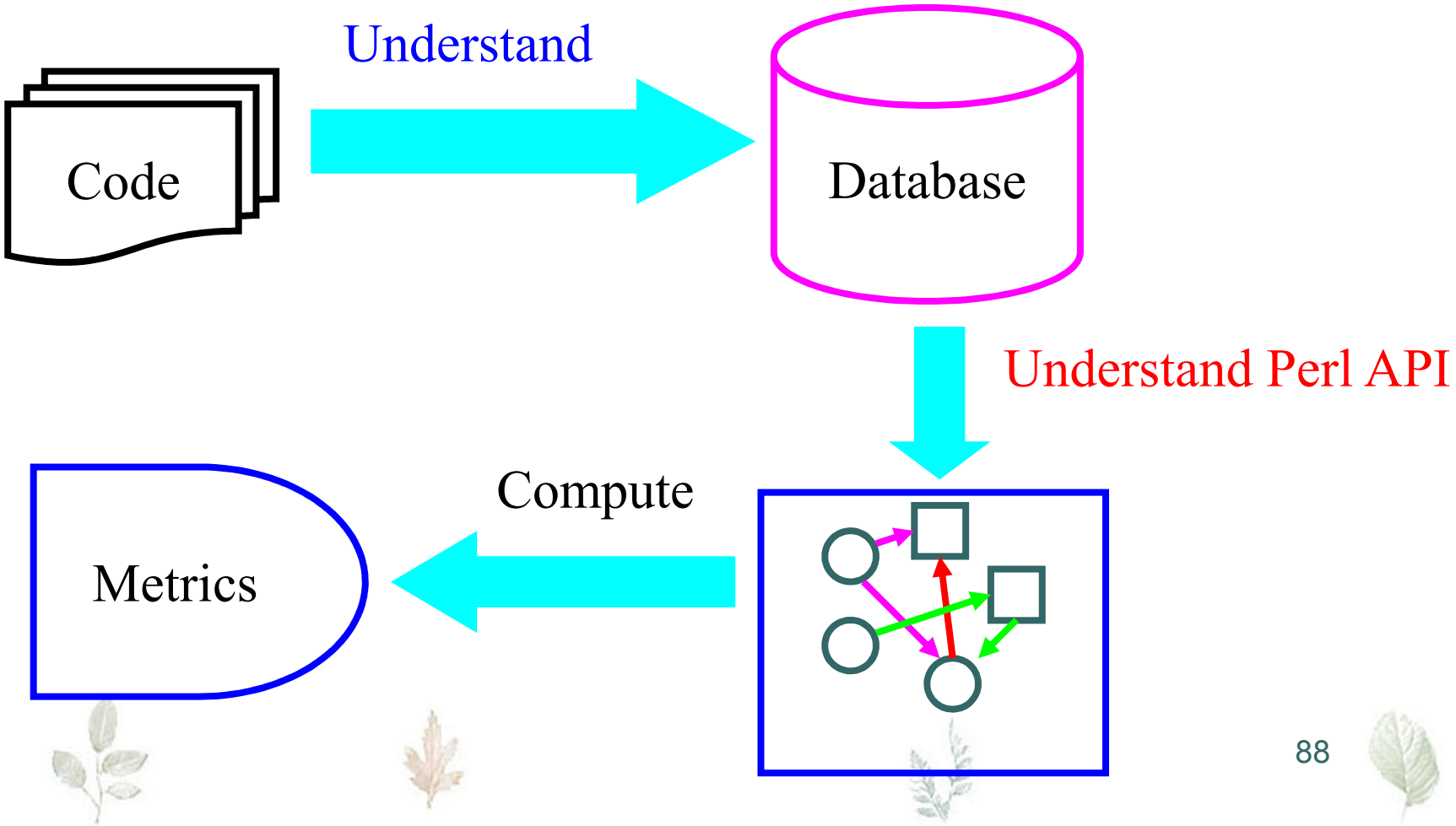




87

Understand: Metrics

2. Custom Metrics



Understand: Metrics

Perl Script example 1

List of Files

```
use Understand;
$db = Understand::open("test.udb");
foreach $file ($db->ents("File")) {

  # print the long name (ie, show directory names)
  print $file->longname(),"\n";
}
```

Understand: Metrics

Perl Script example 2

Cyclomatic Complexity of Functions

```
use Understand;
$db = Understand::open("test.udb");

# lookup a specific metric
foreach $func ($db->ents("Function")) {
  $val = $func->metric("Cyclomatic");

  # only if metric is defined for entity
  print $func->name()," = ",$val,"\n" if
 defined($val);
}
```

Understand: Metrics

Perl Script example 3

Called By Graphs of Functions

```
use Understand;
 $db = Understand::open("test.udb");

 # loop through all functions
 foreach $func ($db->ents("Function")) {
   $file = "callby_" . $func->name() . ".png";
   print $func->longname(), " -> ", $file,"\n";
   $func->draw("Called By",$file);
 }
```

Understand: Metrics

Perl Script example 4

Lexical stream

```
use Understand;
$db = Understand::open("test.udb");

$file = $db->lookup("test.cpp");
$lexer = $file->lexer();

foreach $lexeme ($lexer->lexemes()) {
  print $lexeme->text();
  if ($lexeme->ent()) {
    print "@";
  }
}
```

Understand: Metrics

Perl Script example 5

Global Variable Usage

```
use Understand;
$db = Understand::open("test.udb");
foreach $var ($db->ents("Global Object ~Static")) {
  print $var->name(),":\n";
  foreach $ref ($var->refs()) {
    printf "  %-8s %-16s %s (%d,%d)\n",
      $ref->kindname(), $ref->ent()->name(),
      $ref->file()->name(), $ref->line(),
      $ref->column();
  }
  print "\n";
}
```

Understand: Metrics

https://scitools.com/support/perl-api/

scitools Understand

support@scitools.com Search

Home Features Buy Support Customers Contact Blog

Download Free Trial

Current Users Download Here

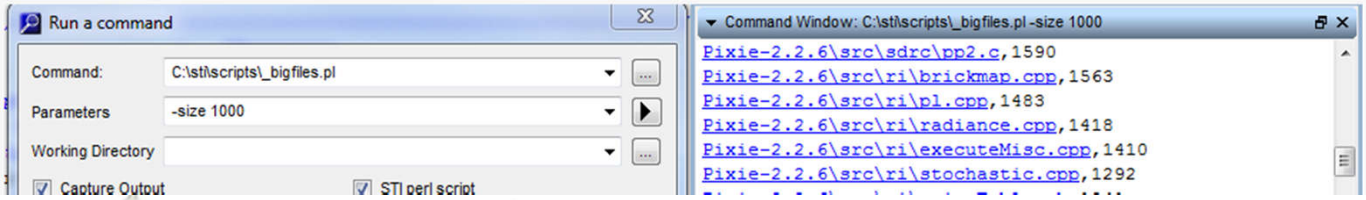
Support > API/Plugins > Perl API

Perl API

Understand includes a PERL API which allows you to directly query the Understand database. If you want to create your own report, or gather information in a manner that we didn't foresee in Understand, you can access the information yourself via the API. The API can be accessed in several different ways: most of the scripts are designed to be run from Understand's GUI on the currently opened database. They can also be run from the command line, and some plugins are designed to interact directly with Understand.

Run Scripts in Understand

To run a script from inside of Understand, use Tools->Run a Command, configure any necessary parameters, and hit run. The resulting output can then be double-clicked to visit the referenced source.



Understand: Metrics

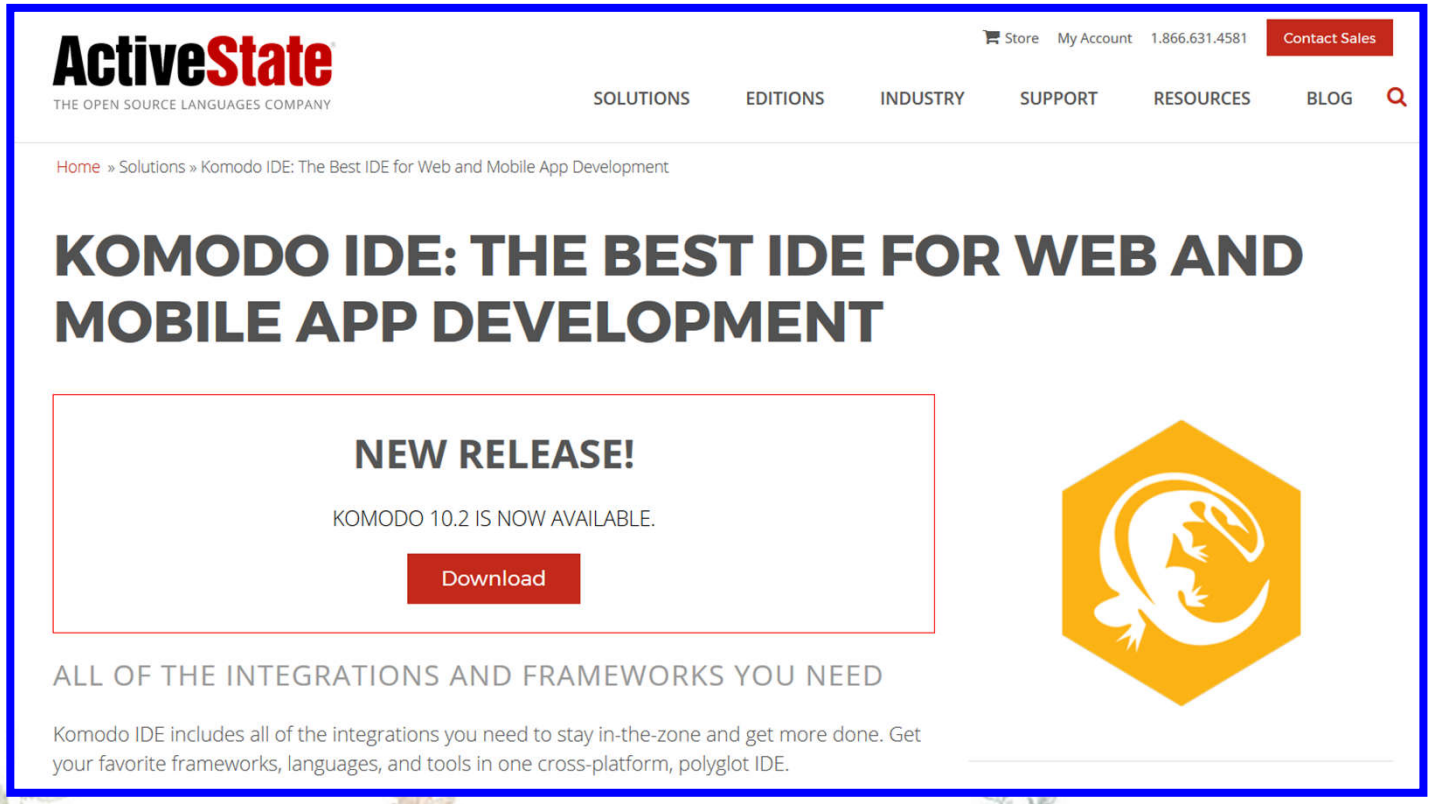
More detailed example scripts are shipped with Understand in the SciTools/scripts/perl folder.

① Report entities that are defined more than once
② Report variables that are not initialized
③ Report unused functions
④ Lists all global variables and where they are defined
⑤ Report duplicate code
⑥ Report dead code
⑦ …

Understand: Metrics

ActiveState Komodo IDE

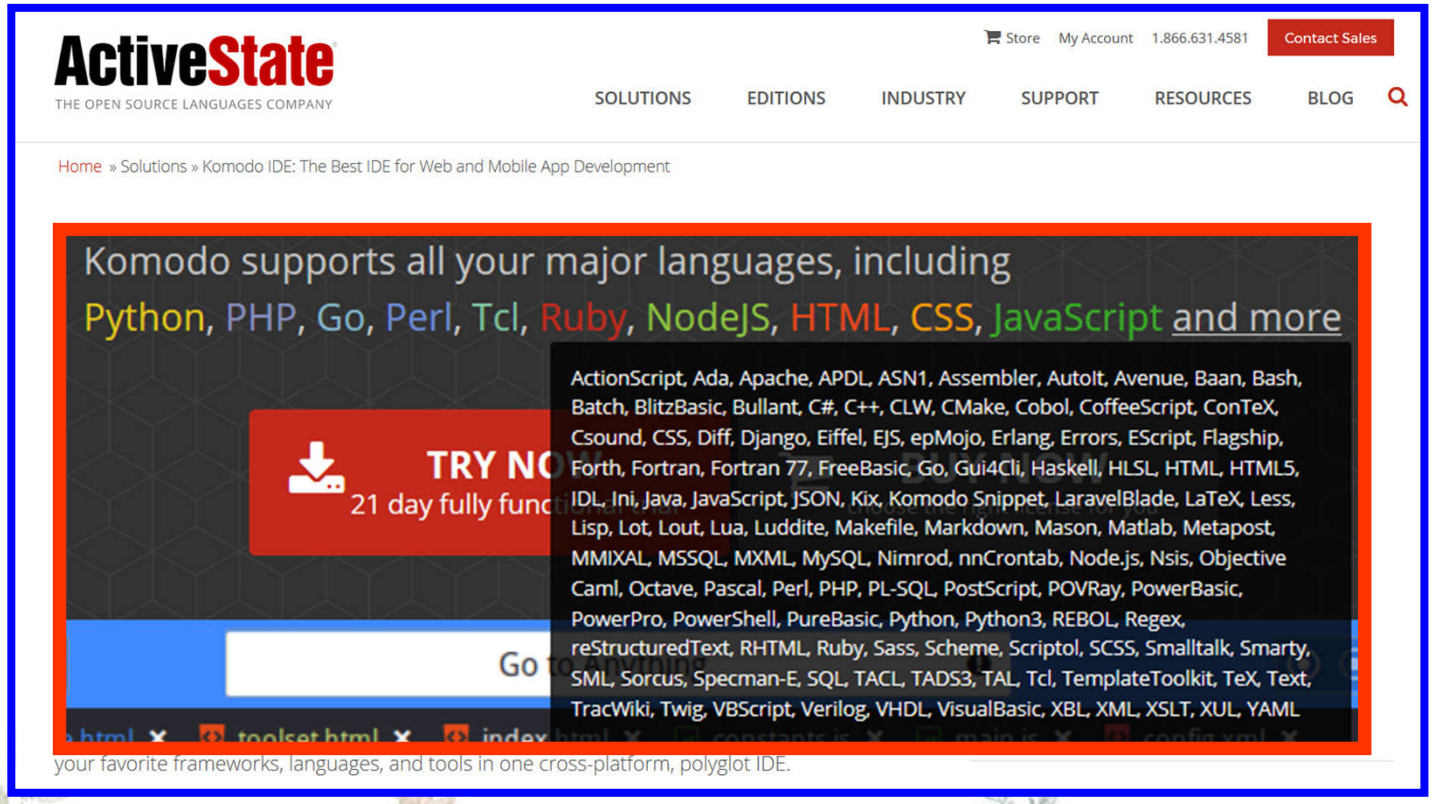
http://komodoide.com/



Understand: Metrics

ActiveState Komodo IDE

http://komodoide.com/



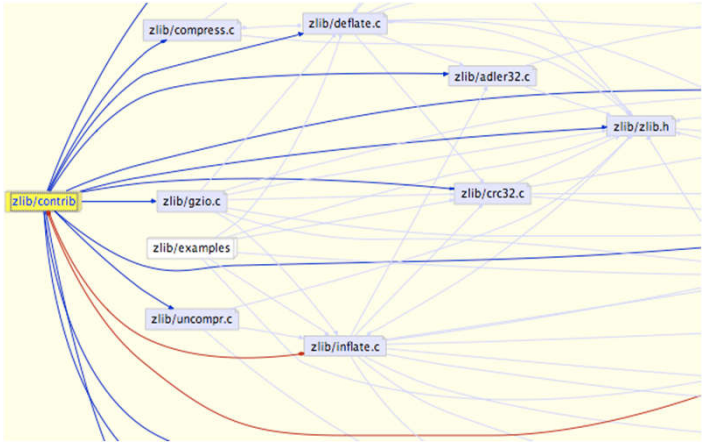
Understand: Metrics

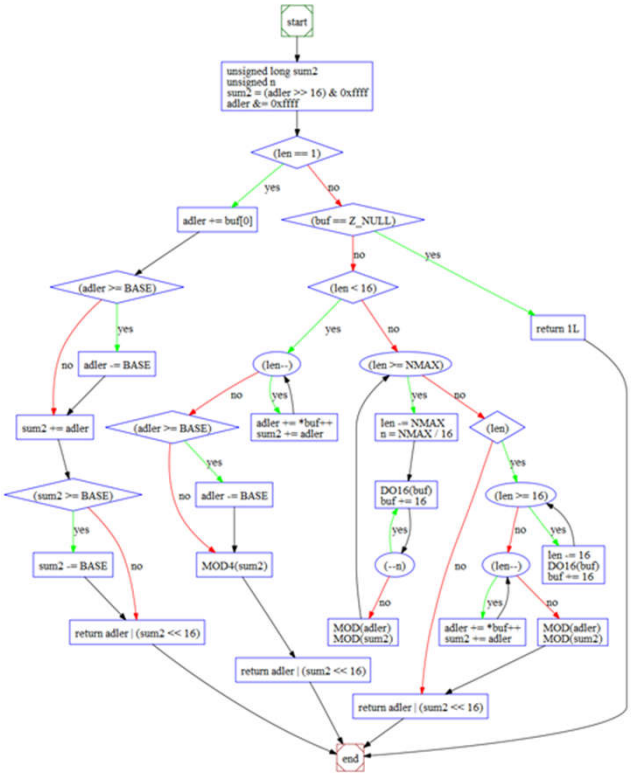
Supported Languages

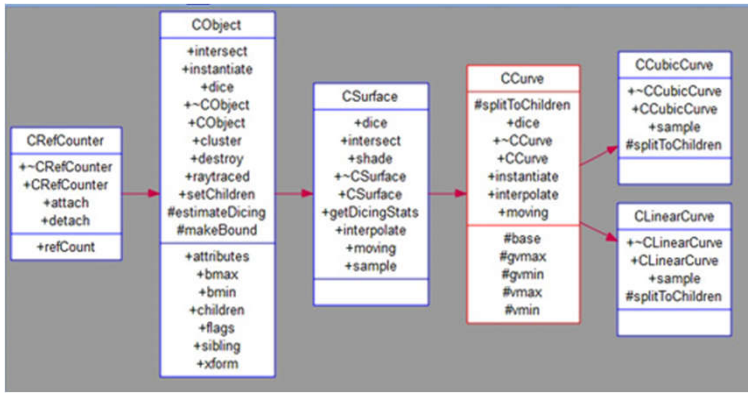
- Ada
- COBOL
- Coldfire 68k Assembly
- C/C++
- C#
- Fortran
- Java
- Jovial
- Pascal
- PL/M
- Python
- VHDL
- Web languages: PHP, HTML, CSS, JavaScript

Understand：其他功能

- 编码规范检查
- 依赖图抽取







Understand

- A. Koru, et al. Comparing high-change modules and modules with the highest measurement values in two large scale open-source products. IEEE TSE, 2005
- Y. Zhou, et al. Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness. IEEE TSE 2009
- …

CountLineCode

Formula: Code && ! Inactive

Result (for function printHello()): 11

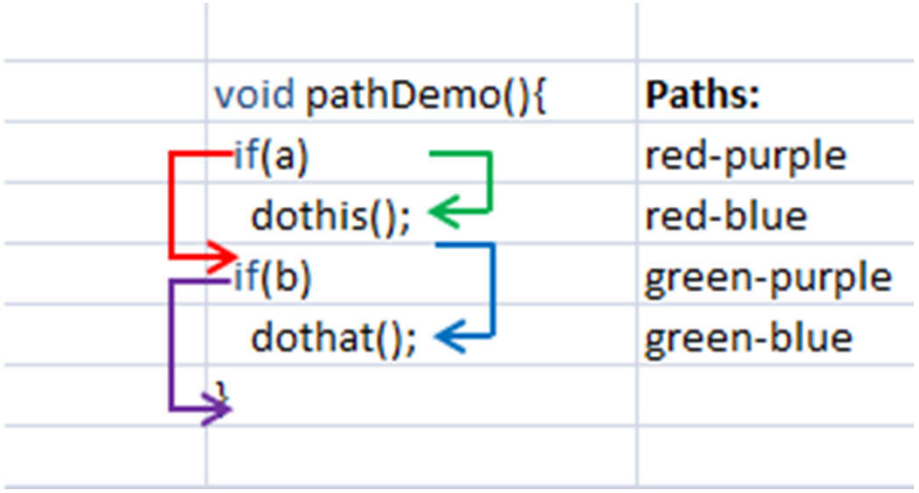
	Code	Comment	Preprocessor	Declarative	Executable	Inactive
void SayHello::printHello(){	1	0	0	1	0	0
switch(i){	1	0	0	0	1	0
case 0:	1	0	0	0	1	0
cout << "Hello World" << endl;	1	0	0	0	1	0
case 1:	1	0	0	0	1	0
cout << "HELLO WORLD!" << endl;	1	0	0	0	1	0
default: //A comment here	1	1	0	0	1	0
for(int m=0; m < j; m++);	1	0	0	1	1	0
cout << "hello world" << endl;	1	0	0	0	1	0
}	1	0	0	0	0	0
#ifdef A_VERY_NICE_VARIABLE	0	0	1	0	0	0
	0	0	0	0	0	1
cout << "Inactive Line" << endl; // Inactive	1	1	0	0	0	1
#endif	0	0	1	0	0	0
	0	0	0	0	0	0
}	1	0	0	0	0	0




CountPath

Formula: The number of paths through the code. This can grow really fast since each if doubles the number of paths through the code.

Result (for function pathDemo): 4



MaxNesting
Formula: MAX(nesting)
Result (for function cyclomaticDemo()): 4

	Cur Nest
void cyclomaticDemo(){	0
bool a=true,b=true,c=true;	0
	0
if(a || (b && c)){	1
while(a? b : c){	2
for(int i=0; i < 10; i ++){	3
switch(i){	4
case 1:	4
case 2:	4
cout<<i<<endl;	4
break;	4
case 5:	4
break;	4
default:	4
cout<<i<<endl;	4
}	4
}	3
}	2
} else{	1
try{	1
do{	2
cout<<a<<b<<c<<endl;	2
}while(a);	2
}	1
catch(...){	1
	1
}	1
}	1
}	0

CountInput

Entity Kind: "c global object, c local object, c member object, c parameter, c function"
Reference Kind: "c use ~ptr ~inactive, c callby ~inactive"
Notes: recursive function calls and local variables that are not class static variables are not included.
Result (for class function inOutFunc): 6

Referenced Entity	Entity Kind	Reference Kind
metrictestknots.h	C Code File	C Definein
in1	C Parameter	C Define
in2	C Parameter	C Define
inout1	C Parameter	C Define
inout2	C Parameter	C Define
out1	C Parameter	C Define
out2	C Parameter	C Define
in	C Object Global	C Use
in1	C Parameter	C Use
in2	C Parameter	C Use
inout1	C Parameter	C Deref Use
inout2	C Parameter	C Use
out	C Object Global	C Set
in1	C Parameter	C Use
inout1	C Parameter	C Deref Set
in2	C Parameter	C Use
inout2	C Parameter	C Set
in1	C Parameter	C Use
out1	C Parameter	C Deref Set
in2	C Parameter	C Use
out2	C Parameter	C Set
somefunc	C Unknown Function	C Call
in1	C Parameter	C Set
in2	C Parameter	C Set

```
int in =1;
int out = 1;

int inOutFunc(int in1, int in2, int *inout1, int & inout2, int * out1, int & out2){
 out = in + in1 + in2 + *inout1 + inout2;

 *inout1 = in1;
 inout2 = in2;

 *out1 = in1;
 out2 = in2;

 in1 = somefunc();
 in2 = 2;

 int randomint = 3;
 in1 = randomint;

 return 4;
}

void callingfunc(){
 int a,b,c,d;
 int myval = inOutFunc(1,2,a,b,c,d);
 int temp = out;
 int l = a + b + c + d;
}
```

These are all repeats of entities already counted above.

CountOutput

Entity Kind: "c global object, c local object, c member object, c parameter, c function"
Reference Kind: "c set, c modify, c call ~inactive"
Notes: recursive function calls, local variables that are not class static variables, and parameters that are pass by value are not included. Also counts non-void return type as 1.
Result (for class function inOutFunc): 7

Referenced Entity	Entity Kind	Reference Kind
metrictestknots.h	C Code File	C Definein
in1	C Parameter	C Define
in2	C Parameter	C Define
inout1	C Parameter	C Define
inout2	C Parameter	C Define
out1	C Parameter	C Define
out2	C Parameter	C Define
in	C Object Global	C Use
in1	C Parameter	C Use
in2	C Parameter	C Use
inout1	C Parameter	C Deref Use
inout2	C Parameter	C Use
out	C Object Global	C Set
in1	C Parameter	C Use
inout1	C Parameter	C Deref Set
in2	C Parameter	C Use
inout2	C Parameter	C Set
in1	C Parameter	C Use
out1	C Parameter	C Deref Set

```
int in =1;
int out = 1;

int inOutFunc(int in1, int in2, int *inout1, int & inout2, int * out1, int & out2 ){
  out = in + in1 + in2 + *inout1 + inout2;

  *inout1 = in1;
  inout2 = in2;

  *out1 = in1;
  out2 = in2;

  in1 = somefunc();
  in2 = 2;

  int randomint = 3;
  in1 = randomint;

  return 4;
}

void callingfunc(){
  int a,b,c,d;
  int myval = inOutFunc(1,2,a,b,c,d);
  int temp = out;
  int l = a + b + c + d;
}
```

Knots

Description: Measure of overlapping jumps. If a piece of code has arrowed lines indicating where every jump in the flow of control occurs, a knot is defined as where two such lines cross each other. The number of knots is proportional to the complexity of the control flow.

https://www.scitools.com/documents/imagesMetrics/KnotsC.png

Cyclomatic Complexity

Description: McCabe Cyclomatic complexity as per the original NIST paper on the subject.
Edges - Nodes + Connected Components
McCabe also proved that the cyclomatic complexity of any structured program with only one entrance point and one exit point is equal to the number of decision points contained in that program plus one (refer to section V for the proof).Understand counts the keywords for decision points (FOR, WHILE, etc) and then adds 1. For a switch statement, each 'case' is counted as 1 and the 'switch' itselfs adds one to the final Cyclomatic Complexity count.

https://www.scitools.com/documents/imagesMetrics/CyclomaticC.png