

Michael J. Sepcot
CS537 – Homework Assignment #1
October 9, 2005

PROBLEM #1 (10 points)

For the relation systems in [Example 2.6](#) in the textbook (pages 34-35), determine which of the following mappings are representations. Explain your answer in terms of the representation condition.

a. $M(\text{each delayed response})=6$; $M(\text{each incorrect output})=6$; $M(\text{each data loss})=69$
This is not a representation of the relation system. The mapping for $M(\text{each delayed response})$ and $M(\text{each incorrect output})$ does not preserve the “is more critical than” relation. We do not have three distinct numbers for our representation.

b. $M(\text{each delayed response})=1$; $M(\text{each incorrect output})=2$; $M(\text{each data loss})=3$
This is a valid representation of the relation system. We have three distinct numbers for our representation that preserve the “is more critical than” relation between each delayed response, each incorrect output, and each data loss.

c. $M(\text{each delayed response})=6$; $M(\text{each incorrect output})=3$; $M(\text{each data loss})=2$
This is not a representation of the relation system. We do have three distinct numbers for our representation, but those numbers do not preserve the “is more critical than” relation between any of the measures of failure.

d. $M(\text{each delayed response})=0$; $M(\text{each incorrect output})=1$; $M(\text{each data loss})=0.5$
This is not a representation of the relation system. The mapping for $M(\text{each data loss})$ does not preserve the “is more critical than” relation because it is mapped to a lower number than the $M(\text{each incorrect input})$ value.

e. $M(\text{each delayed response})=-1$; $M(\text{each incorrect output})=1$; $M(\text{each data loss})=2$
This is a valid representation of the relation system. We have three distinct numbers for our representation that preserve the “is more critical than” relation between the fault mappings.

PROBLEM #2 (10 points)

In the [Example 2.18](#) in the textbook (page 50), determine the affine transformations from:

- a. M_1 to M_2 $M_2 = (M_1 - 1) * 2$
- b. M_2 to M_1 $M_1 = (M_2 / 2) + 1$
- c. M_2 to M_3 $M_3 = M_2 + 3.1$
- d. M_3 to M_2 $M_2 = M_3 - 3.1$
- e. M_3 to M_1 $M_1 = (M_3 - 1.1) / 2$

PROBLEM #3 (20 points)

Determine which of the following statements are meaningful or meaningless (for each statement **justify** your answer):

- a. The average size of a Windows application program is about four times that of a similar Unix program.

This statement is meaningful. Program size is generally measured in lines of code which is measured on a ratio scale and can be compared in such a way.

b. An average number of defects in Java programs is twice as high as an average number of defects in Assembler programs.

This statement is meaningful. Because we are counting entities, we are making measurements on an absolute scale and average number of the two sets can be compared in such a way.

c. A program written in C++ is on average 850 lines shorter than a functionally equivalent program written in Assembler.

This statement is meaningful. We are measuring the length of programs on a ratio scale which can be compared in such a way.

d. A program written in Assembler is on average 1.5 longer than a functionally equivalent program written in C++.

This statement is meaningless. The term "longer" can refer to either the time it took to program or the number of lines of code in the program. Because of the ambiguity of the statement, it is meaningless.

e. The quality of program A is higher than program B.

This statement is meaningful. Quality of a program is an ordinal measure which can be compared in this manner.

f. The effort of producing program A is twice that of producing program B.

This statement is meaningless. Effort of producing a program can be measured in a wide variety of ways (cost, man-months, etc.). Because of the ambiguity of measurement, this statement is meaningless.

g. Software A is more reusable than software B.

This statement is meaningless. Can the software A be used verbatim in more projects than software B, or does software A have more reusable functions/modules than software B, are we considering reuse verbatim or with slight modification? With all of these open questions, the ambiguity of measurement makes this statement meaningless.

h. An average temperature in July is twice as high as an average temperature in March.

This statement is meaningless. The scale of measurement needs to be provided before this statement can have meaning.

i. In Chicago, an average temperature in August is 30°F lower than an average temperature in March.

This statement, while false, is meaningful. We have a scale of measurement indicated which allows us to make comparisons between averages.

j. The combined size of test suites TS_1 and TS_2 is smaller than the size of test suite TS_3 .

This statement is meaningful. Size of software is measured in lines of code and can be compared in this manner.

PROBLEM #4 (25 points)

Given the following set of empirical relations for attribute *component testability*:

R1: "is very highly testable component"

R2: "is highly testable component"

R3: "is moderately testable component"

R4: "is hardly testable component"

R5: "is more difficult to test than"

Our understanding of this relation (R5) is that:

- a. all moderately testable components *are more difficult to test than* all highly testable components.
- b. all moderately testable components *are more difficult to test than* all very highly testable components.
- c. all highly testable components *are more difficult to test than* all very highly testable components.
- d. all hardly testable components *are more difficult to test than* all very highly testable components.
- e. all hardly testable components *are more difficult to test than* all highly testable components.
- f. all hardly testable components *are more difficult to test than* all moderately testable components.

It is assumed that the classification (R1, R2, R3, and R4) is mutually exclusive and jointly exhaustive.

Prove using the Cantor's theorem that the scale for the set of relations is an ordinal scale.

From *Software Metrics* we know that 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 is acceptable
- The numbers represent ranking only

Cantor's theorem states that the empirical relation system has an ordinal scale representation if and only if R is a strict weak order. R being strict weak order means that the relation is:

- Asymmetric – meaning that if x is related to y then it follows that y is not related to x
- Negatively Transitive – meaning that if x is related to y, then for every z, either x is related to z, or z is related to y

R5 states that x "is more difficult to test than (>)" y. This relationship hold true for the classification (R1, R2, R3, and R4) such that $R4 > R3 > R2 > R1$. The classification is mutually exclusive and jointly exhaustive which makes this relationship asymmetric and negatively transitive.

PROBLEM #5 (15 points)

Consider the attribute *component testability* of Problem #4.

- a. Refine the empirical system so that the scale type is interval.

$M(x) =$

- 1 if x is a very highly testable component
- 2 if x is a highly testable component
- 3 if x is a moderately testable component
- 4 if x is a hardly testable component

The mapping (M) provided above preserves order and preserves differences in the empirical system. Any other mapping will work for this empirical system provided there is an admissible transformation such that $M = aM' + b$.

- b. Refine the empirical system so that the scale type for the component testability is ratio.

0 if x does not need to be tested
1 if x is a very highly testable component
M(x) = 2 if x is a highly testable component
3 if x is a moderately testable component
4 if x is a hardly testable component

The mapping (M) provided above will work for the ratio scale. The mapping preserves the order, the size of intervals between entities, and the ratios between entities in the empirical system. A zero element was added to represent the total lack of the attribute. And the measurement mapping starts at zero and increases at equal intervals. Any other mapping will work for this empirical system provided there is a ratio translation such that a positive scalar, a, fulfills: $M = aM'$.

PROBLEM #6 (20 points)

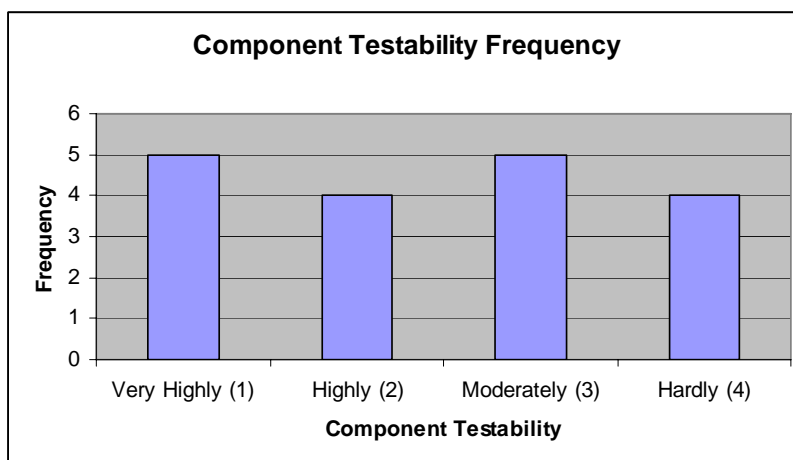
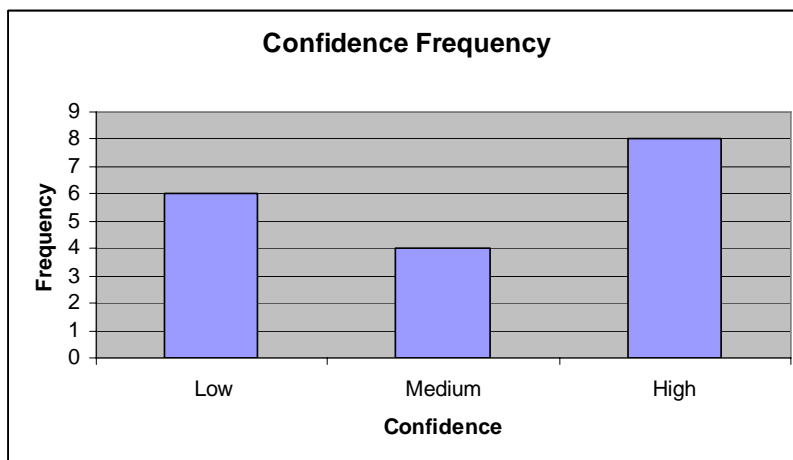
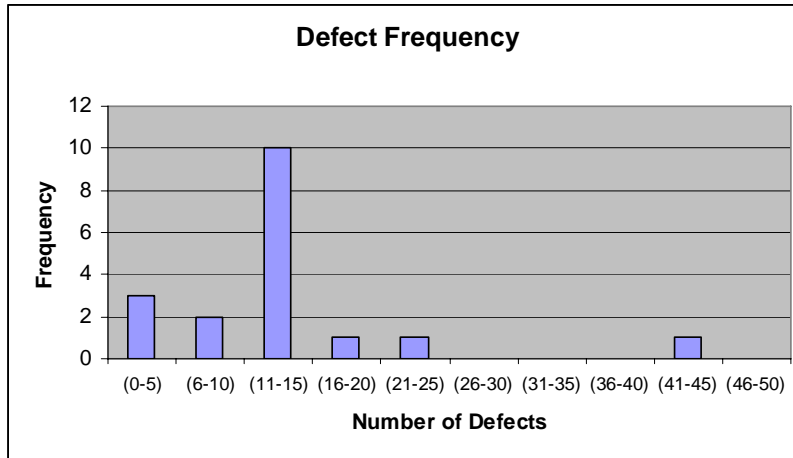
Given the following data collected in a hypothetical software company:

<u>Component</u>	<u>LOC</u>	<u>Component testability</u>	<u>Confidence</u>	<u># of Defects</u>
A	550	1	Low	18
B	149	4	Low	12
C	545	2	High	0
D	247	3	Medium	25
E	409	1	Low	15
F	256	4	High	12
G	512	3	Low	15
H	125	2	High	12
I	1256	3	High	12
J	170	1	Medium	2
K	420	3	High	12
L	410	1	High	10
M	245	2	Medium	12
N	153	4	High	42
O	567	1	Medium	14
P	235	4	High	12
Q	634	2	Low	3
R	20	3	Low	7

Where LOC: Lines Of Code

For each of the following attributes *LOC*, *Component Testability*, *Confidence*, and *# of Defects* determine:

- a. distribution frequencies (there should be at least three categories),



b. an arithmetic mean,

Lines of Code: 383.500
Component Testability: 2.444
Confidence: Cannot Compute. The values are not on the ordinal scale.
Number of Defects: 13.056

c. a mode,

Lines of Code: All items occurred equal number of times (once)
Component Testability: Very Highly (1), Moderately (3)
Confidence: High
Number of Defects: 12

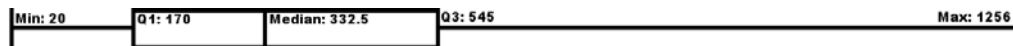
d. a median,

Lines of Code: (247, 256)
Component Testability: (2, 2)
Confidence: Cannot Compute. The values are not on the interval scale.
Number of Defects: (12, 12)

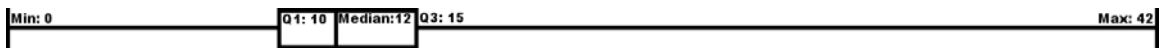
e. a box plot (draw box plots only for *LOC* and *# of Defects*)

Key: [Min, Q1, Median, Q3, Max]

Lines of Code – Box Plot [20, 170, 332.5, 545, 1256]



Number of Defects – Box Plot [0, 10, 12, 15, 42]



Use the set of empirical relations shown in Problem #4 for the attribute *component testability*.

The following values have been assigned to the *component testability*:

- 1 – Very highly testable component
- 2 – Highly testable component
- 3 – Moderately testable component
- 4 – Hardly testable component