Foundations

of

Software Measurement

Horst Zuse

August 22, 1995

Contents

1 Introduction

Maintenance of Software

- Today, we have still the fact that over 70% of the software development effort is spent in testing and maintenance of software.

- Schedule and cost estimates of software are grossly inaccurate, software has still a poor quality and the productivity rate for software is increasing more slowly than the demand for software.

- Schneidewind points out that there exists a maintenance problem because: 70 - 80% percent of the existing software was produced prior to significant use of structured programming.

- It is difficult to determine whether a change in code will affect something, It is difficult to relate specific programming actions to specific code.

- The major problem in doing maintenance is that we cannot do maintenance on a system which is not designed for maintenance.

Introduction (Cont.)

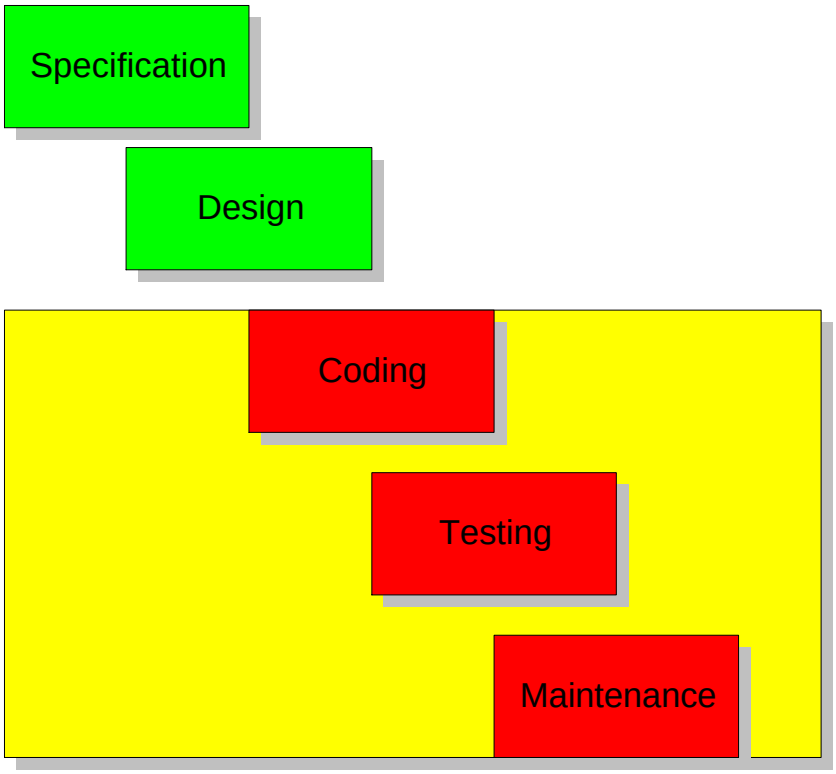


Figure 1: Software Life-Cycle. The expensive phases are coding, design, testing and maintenance phases.

1 Introduction (Cont.)

- These facts have generated the need of determining the quality of software with engineering methods.

- For this reason, computer scientists and engineers have begun to place increasing attention on quantitative methods as an information source of the quality of software.

- Already in 1976, Belady and Lehman stated in their classic research on the development of the IBM OS/360 operating system /BELA76/: *Law of increasing entropy: the entropy of a system (its unstructuredness) increase with time, unless specific work is executed to maintain or reduce it. Entropy can result in severe complications when a project has to be modified and is generally an obstacle of maintenance.*

- These statements above characterize the need of quantitative methods in order to understand the function of a large software system.

Introduction (Cont.)

Structure Chart

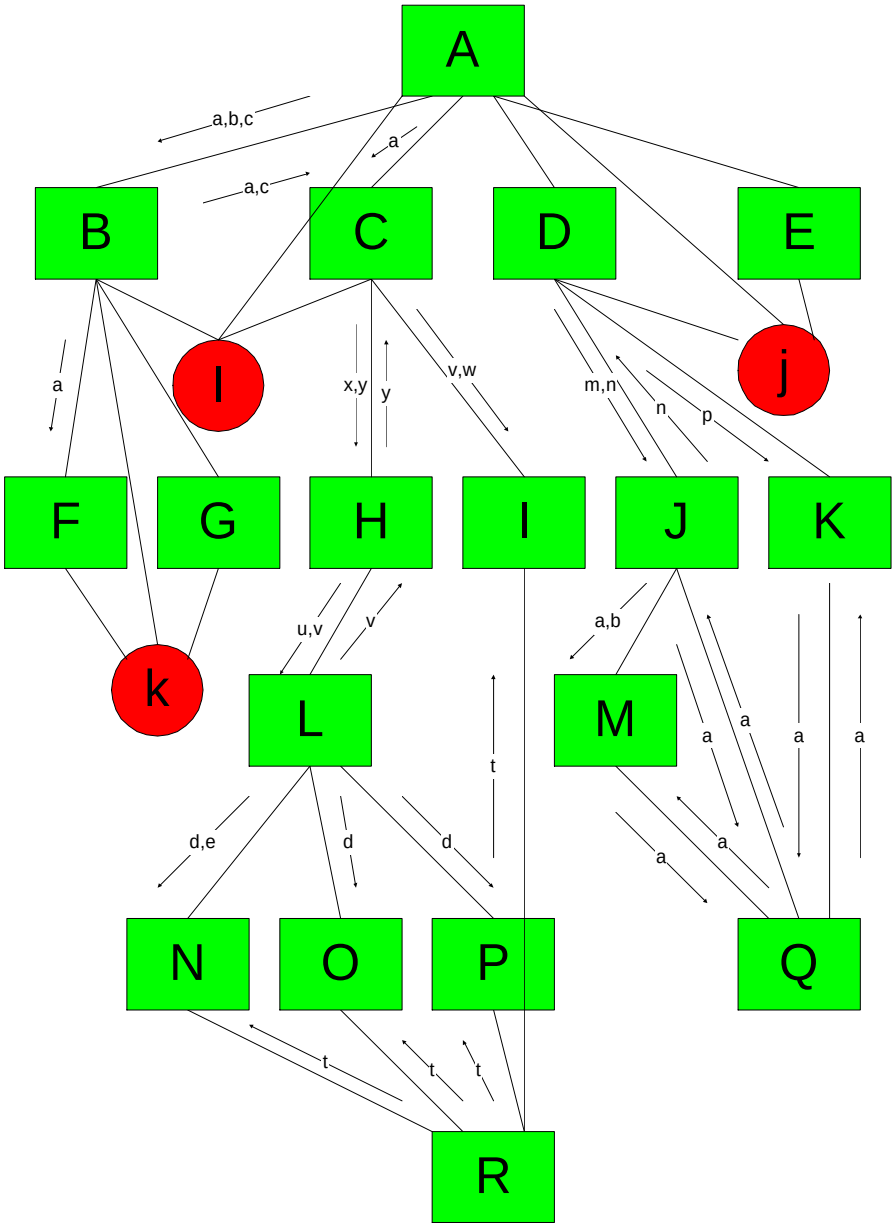
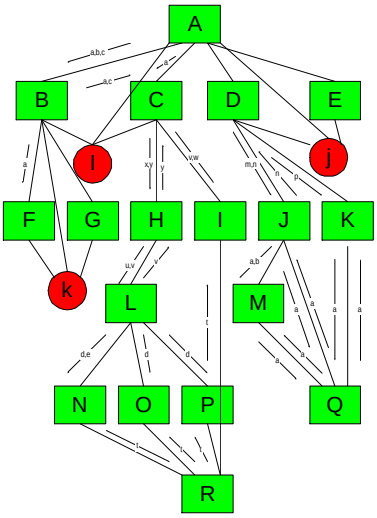


Figure 2: Structure Chart.

The question is what components are difficult to maintain.

Introduction (Cont.)

Structure Chart and an Analysis with A Software Measure



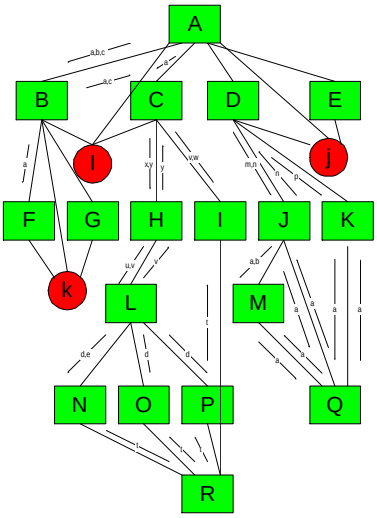
STRUCTURE CHART.....: D-BM
COMMUNICATIONS BETWEEN THE MODULES:

#	M1	M2	PARAM	GLOB	SC(M1,M2)
1	A	B	3	1	5
2	A	C	2	1	4
3	A	D	0	1	2
4	A	E	0	1	2
5	B	C	0	1	2
6	B	F	1	1	3
7	B	G	1	1	3
8	C	H	2	0	2
9	C	I	2	0	2
10	D	E	0	1	2
11	D	J	2	0	2
12	D	K	1	0	1
13	F	G	0	1	2
14	H	L	2	0	2
15	I	R	1	0	1
16	J	M	2	0	2
17	J	Q	1	0	1
18	K	Q	1	0	1
19	L	N	2	0	2
20	L	O	1	0	1
21	L	P	1	0	1
22	M	Q	1	0	1
23	N	R	1	0	1
24	O	R	1	0	1
25	P	R	1	0	1

Figure 3: External and data coupling between modules.

Introduction (Cont.)

Structure Chart and an Analysis with a Software Measure




```
STRUCTURE CHART.....: D-BM
AGGREGATED COMPLEXITIES:
#     MODULE          IMOD   IS   AS    AGGR--MODULES
-------------------------------------------------------
    1 A                  5   13   39   A B C D E
    2 B                  4   11   25   B C F G
    3 C                  3    8   15   C H I
    4 D                  4    5   14   D E J K
    5 E                  1    2    2   E
    6 F                  2    3    6   F G
    7 G                  1    3    3   G
    8 H                  2    4   10   H L
    9 I                  2    3    4   I R
   10 J                  3    5    9   J M Q
   11 K                  2    2    3   K Q
   12 L                  4    6   13   L N O P
   13 M                  2    3    4   M Q
   14 N                  2    3    4   N R
   15 O                  2    2    3   O R
   16 P                  2    2    3   P R
   17 Q                  1    1    1   Q
   18 R                  1    1    1   R

SUM OF AGGREGATED COMPLEXITY AS:          77.00
AVERAGE AGGREGATED COMPLEXITY  :           2.61
MINIMAL AGGREGATED COMPLEXITY  :              1
MAXIMAL AGGREGATED COMPLEXITY  :             39
```

Figure 5: Does the Measure of Bowles correspond with the intuitive feeling?

Introduction (Cont.)

Structure Chart and an Analysis with a Software Measure

IMOD	Number of modules connected by data and external coupling.M
IS	Complexity of the module based on parameters and global variables.
AS	Aggregated Complexity of the module (Ripple effect complexity)
AGGR	Modules which have to be maintained if Module # is modified.

Object-Oriented Measurement

- Do object-oriented techniques make the design more robust, more maintainable, more understandable, or more reuseable?

- Is measurement in the object-oriented area not necessary?

- This is a poignant question, since there have been many recent examples where applications designed with so-called OO-methods have turned out not to fulfil those claims /MART94/.

- Are these qualities of robustness, maintainablity, reuseablility intrinsic to object-oriented design (OOD)? If so, why do not all applications designed with OOD have them. If not, then what other characteristics does an OOD require in order to have these desirable qualities?.

- It is a fact, that at this time more than 200 software measures in the object-oriented area were proposed. These measures are used for analyzing understandability.

Introduction (Cont.)

Simple Measures

Measure LOC

LOC = |N|

Measures of McCabe 1976

MCC-V = |E| - |N| + 2 or MCC-V2 = |E| - |N| + 1.

Measure Defect Density

DD = Defects / LOC

Measure of Henry et al. 1981

D-INFO = Σ (fi(i) * fo(i))^2

COCOMO-Model 1981

EFFORT = a LOC^b , a, b >0.

Function-Point Method

FP = UC * (0.65 + 0.01, $\sum_{i=1}^{14} Fi$, where

UC = 4 I + 5 O + 4 E + 10 L + 7 F., and

TCF = 0.65 + 0.01$\sum_{i=1}^{14} Fi$,

Simple Measure in the Object-Oriented Area

Chidamber and Kemerer 1991 / 1994

CBO = Coupling between Objects

DIT = Depth of the Inheritance Tree

LCOM = Lack of Cohesion

NOC = Number of Children

Other Measures: Abreu, Henderson-Sellers, etc.

Validation and Prediction

The acceptance of software measures by practioneers depends on whether a software measures is valid or can be used as a predictor.

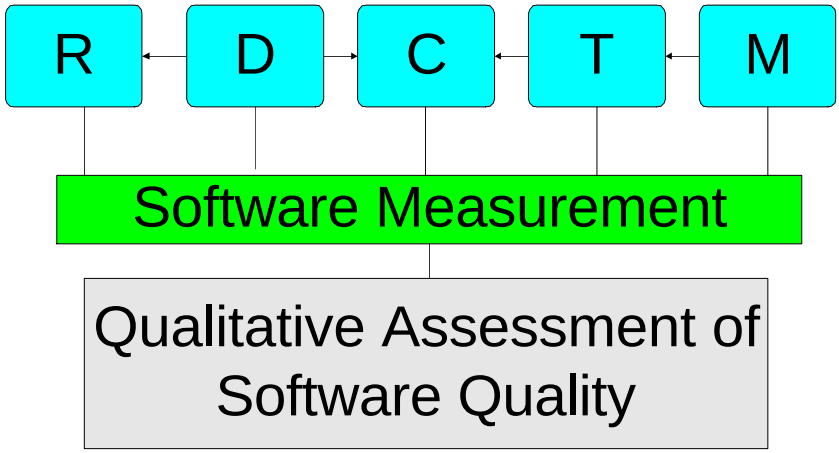


Figure 5: Qualitative Assessment during the Software Life-cycle.

Validation and Prediction

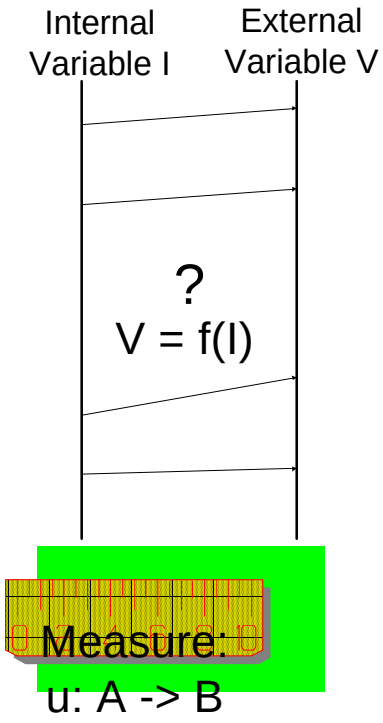


Figure 5: Internal and External Variables.

Internal Variable:	Measure u
External Variable:	Costs of maintenance, etc.
Function f	Function, which has to be validated.

Question: What properties of an external variable can be predicted?

Software Measures

- How can the results of software measurement (product and processes) be interpreted.

- Is it possible to predict the error-proneness of a system using software measures from its design phase.

- Is it possible to extract quantitative features from the representation of a software design to enable us to predict the degree of maintainability of a software system.

- Are there any quantifiable key features of the program code of a module that would enable us to predict the degree of difficulty of testing for that module, and the number of residual errors in the module after a particular level of testing has occurred.

Software Measures (Cont.)

- Is it possible to extract quantifiable features from the representation of a software design to enable us to predict the amount of effort required to build the software described by that design.

- What properties of software measures are required in order to determine the quality of a design..

- Are there features in order to predict the size of a project from the specification phase.

- What are appropriate software measures to underlie the software quality attributes of the ISO 9126 norm by numbers.

- Above we showed that one goal of using software measures is to predict error-proneness or maintainability risk. Following Berns /BERN84/ it holds: *the more difficult to maintain, the higher its maintainability risk.*

Software Measures (Grady 1992)

1. We use them to derive a basis for estimates,
2. to track project progress,
3. to determine complexity,
4. to help us to understand when we have achieved a desired state of software quality,
5. to analyze our defects,
6. and to experimental validate best practices.
7. **In short**, they help us to make better decisions.

Numbers, Numbers, Numbers

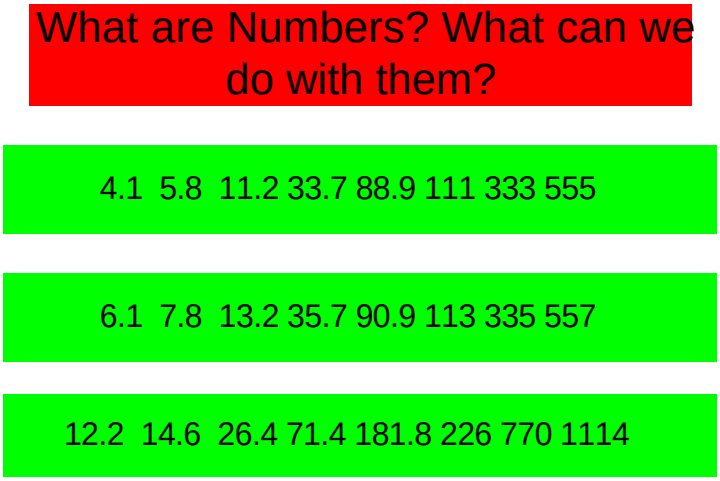


Figure 6: What is the meaning of Numbers?

- We know that 333 is greater than 111 and we know that the number 333 is three times greater than 111.
- We also know, that we can add numbers, for example 111 + 333 = 444.
- Comparing the first row with the second row we see that the numbers in the second row are always by two greater.
- The third row is a multiplication of two of the second row.
- The question is whether these are only mathematical modifications are do we have other objects. Only considering the numbers, we cannot decide that. We need criteria of the properties of the numbers.

2 Measurement Theory (Definition by Roberts)

Measurement has something to do with assignment of numbers that correspond to or represent or preserve certain objected relations.

In the case of temperature, measurement is the assignment of numbers that preserve the observed relation warmer than. In the case of mass, the relation preserved is the relation heavier than. More precisely, suppose A is a set of objects and the binary relation a W b holds if and only if you judge a to be warmer than b.

Then we want to assign a real number f(a) to each $a \in A$ such that for all a, $b \in A$,

$$a \; W \; b \iff f(a) > f(b).$$

The statement above is the base of measurement. Without this statement we cannot get a ranking order of objects. Later, we denote W as an empirical relation and write $\succ$ for *warmer or* $\succeq$ *for equally warm or warmer.*

Software Quality Factors

In the software measurement field, quality factors and subfactors have to be expressed by measures.

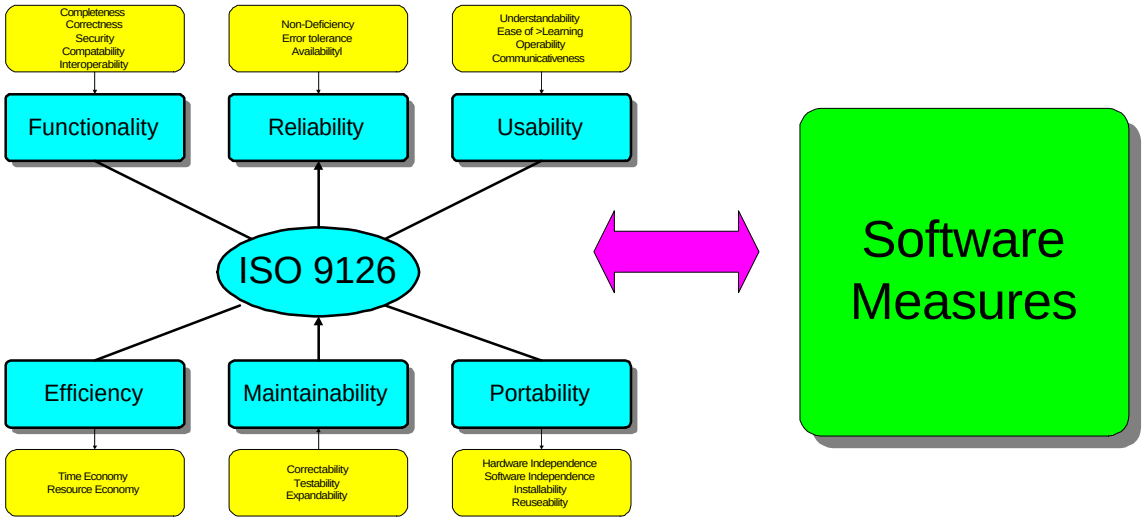


Figure 7: Software quality model ISO9126 with quality factors and subfactors, which have to be expressed by software measures.

Software Quality Factors (Cont.)

For example, software measures for the factor Maintainability and the subfactor Expandability are change counts, like change effort, change size, change rate, change count.

More precisely, suppose P is a set of objects and the binary relation $a \bullet\geq b$ holds if and if only you judge the maintainability of a is greater than for b. Then we want to assign a real number f(a) to each $a \in P$ such that for all $a, b \in P$,

$$a \bullet> b <=> f(a) > f(b).$$

If we include the binary relation *equally or more difficult to maintain,* then we have the relation $\bullet\geq$ and it holds:

$$a \bullet\geq b <=> f(a) \geq f(b).$$

Empirical Statements

- There are several reasons to consider software measurement from an empirical view. We collected some of them.

- Empirical conditions or statements are more intuitive for the user than mathematical statements.

- In reality empirical statements are considered together with the Measures LOC and the Measures of McCabe, for example the term *difficulty of maintenance*, or *complexity* of a program.

- Quality characteristics of software are also described by empirical statements in the ISO 9126 norm /ISO.91a/.

- Different formal definitions of a measure can be reduced to one empirical condition, for example, wholeness (*the sum should be greater than the sum of the parts*) can be reduced numerically to an additive measure without modifying the empirical assumptions.

- Another example is the normalization of software measures to numbers between 0 and 1.

- The GQM (Goal-Question-Measure paradigm) of Basili et al. /BASI84/ defines goals of measurement empirically, the questions are also empirically, but the measure are formally.

- Discussing measurement scales, like ordinal, interval, ratio or absolute scales, we need both: the empirical and formal conditions.

- The calculation of correlations (for example with Spearman, Kendell Tau or Pearson) between a software measure and an external variable, like costs of maintenance, requires an empirical interpretation of the measure and the external variable. The reason is that the use of correlation coefficients requires certain scales levels. Scales are defined by empirical and numerical relational systems (Scale types are defined by admissible transformations). The correlation coefficient itself is used to compare empirical relational systems.

- The Basic COCOMO-model is defined as: EFFORT = $a\ LOC^b$, where $a, b > 0$.

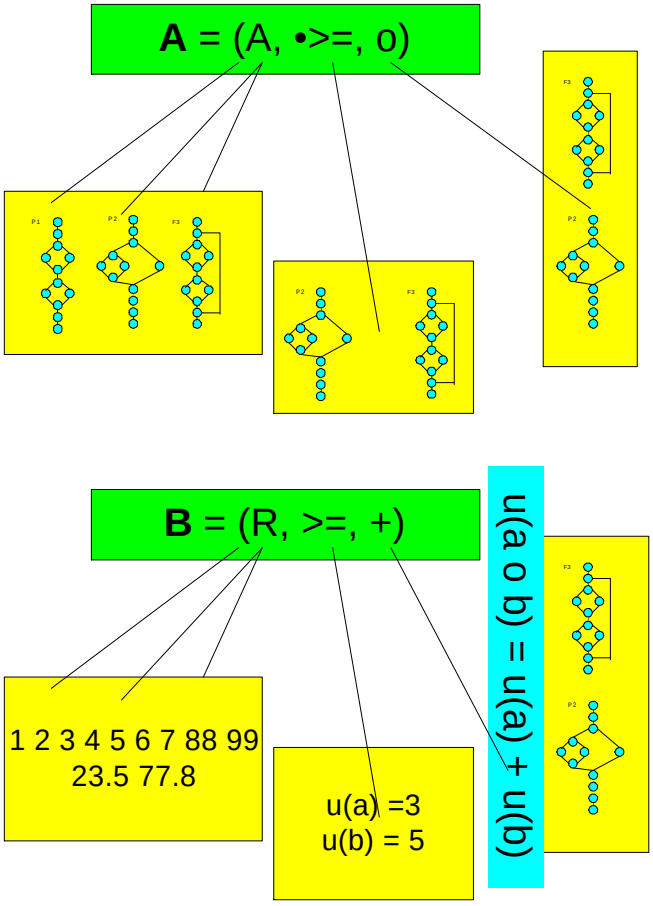
Definition of a Measure

Empirical Relational System

A = (A, •≥, o)

Numerical Relational System

B = (ℜ, ≥, +)



Definition 4.x (Measure):

A measure is a mapping: u: A -> B such that the following holds for all $a,b \in A$:

$$a \bullet\geq b \Leftrightarrow u(a) \geq u(b)$$

Then the Triple (**A**, **B**, u) is called a **scale**.
®

This definition is the basic definition of a measure.

Definition 4.x (Measure):

A measure is a mapping: u: A -> B such that the following holds for all $a,b \in A$:

$$a \bullet\geq b \Leftrightarrow u(a) \geq u(b)$$

and

$$u(a \circ b) = u(a) + u(b)$$

Then the Triple (**A**, **B**, u) is called a **scale**.
®

This definition is an extended definition of a measure which includes the additive homomorphism.

Homomorphism with Flowgraphs

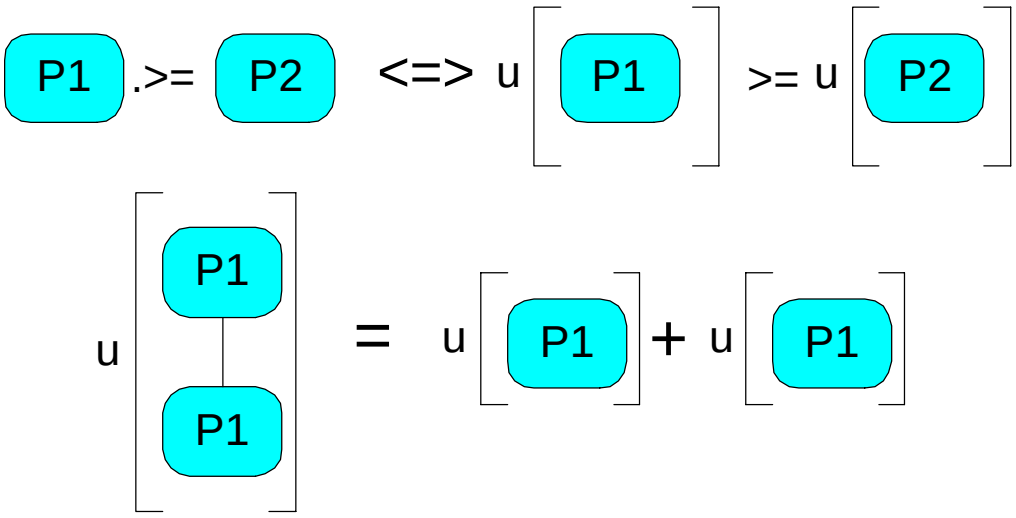


Figure 8: Ranking and additive homomorphism. P1 and P2 are arbitrary flowgraphs.

Additivity for Structure Charts

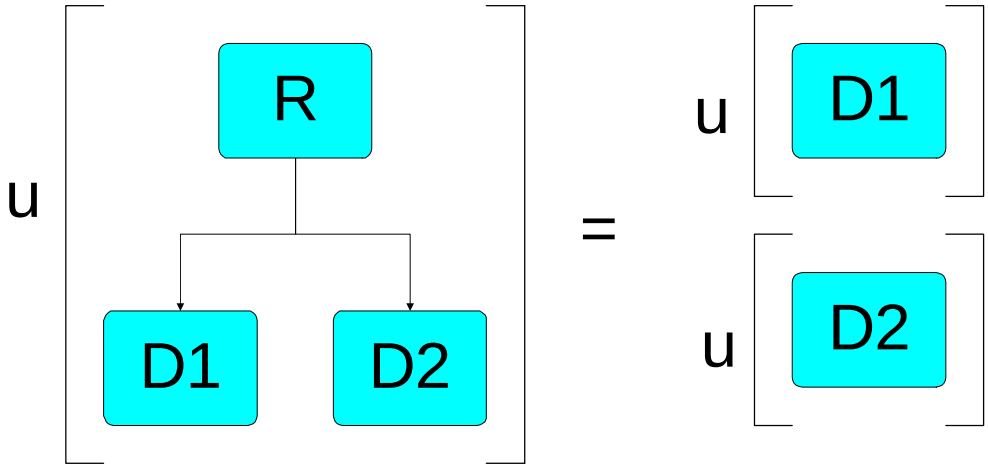
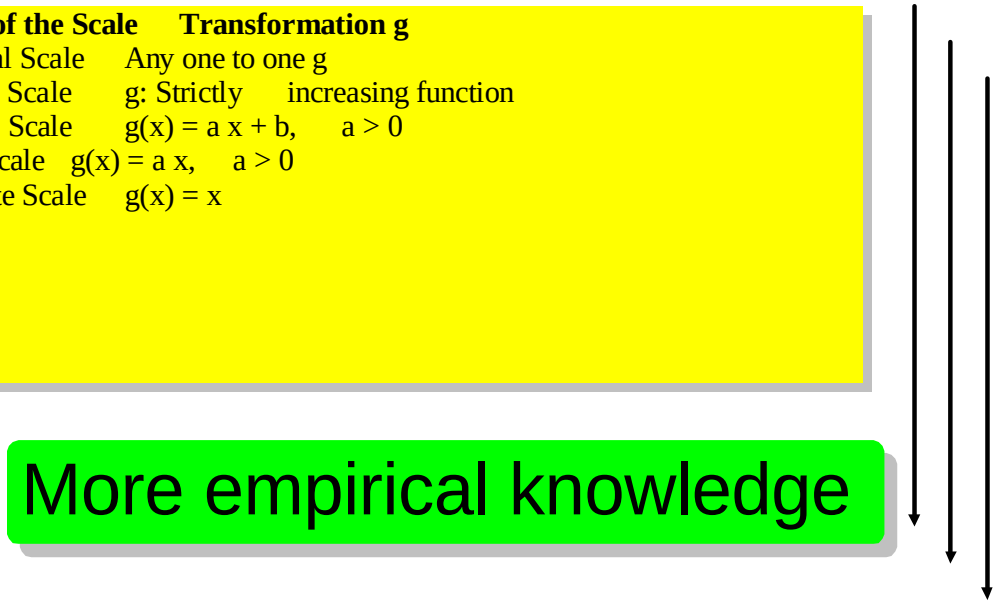


Figure 9: Additive homomorphism of structure charts.

Scale Types

Scale Types are defined by admissible transformations.

Name of the Scale	Transformation g
Nominal Scale	Any one to one g
Ordinal Scale	g: Strictly increasing function
Interval Scale	$g(x) = a x + b, \quad a > 0$
Ratio Scale	$g(x) = a x, \quad a > 0$
Absolute Scale	$g(x) = x$



Scale Types and Scales:

- Scales and Scale types are different things.
- A scale is defined by a homomorphism.
- A scale type is defined by an admissible transformtion.
- Scales are not the power of measurement theory, the power are the empirical and numerical conditions.
- Behind scale types empirical properties are hidden.

- The higher the scale type the more empirical conditions are hidden.

Measurement Process

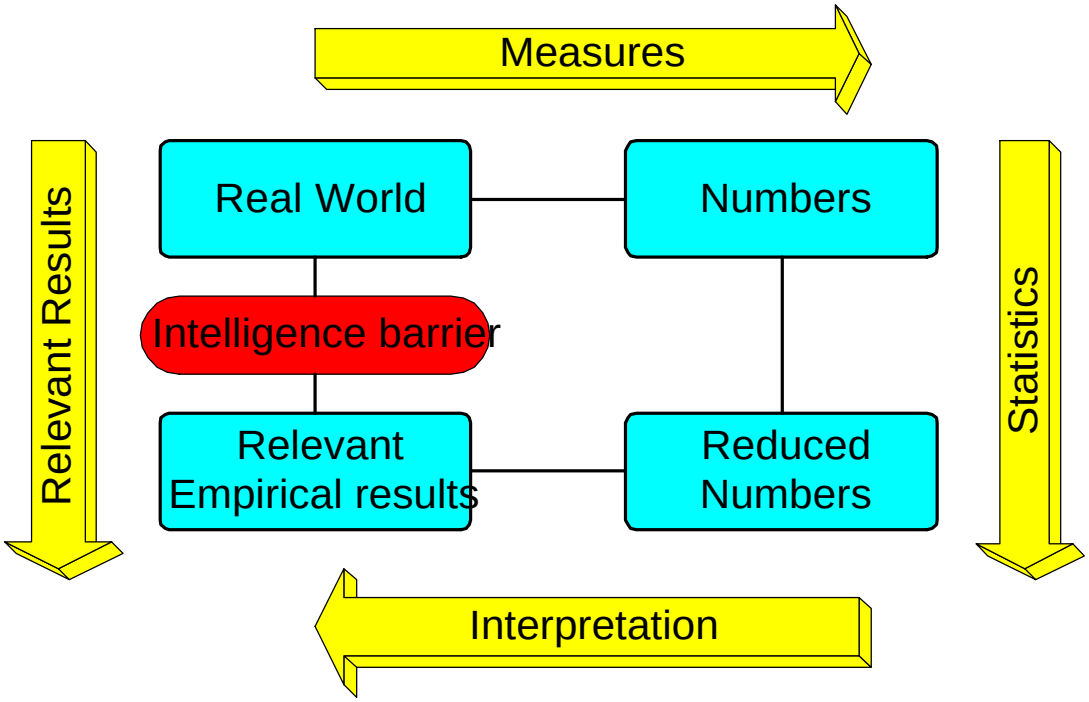


Figure 10: Measurement Process of Kriz.

Theorem of Krantz et al. 1971

Let A be a non empty set, $\bullet\geq$ is a binary relation on A, and o a closed binary operation on A. Then (A, $\bullet\geq$, o) is a closed extensive structure iff there exists a real-valued function on P such that for all a, b $\in$ A:

$$(1) \quad a \bullet\geq b <=> u(a) \geq u(b)$$

and

$$(2) \quad u(a \circ b) = u(a) + u(b)$$

Another function u' satisfies (1) and (2) iff there exists $\alpha > 0$ such that

$$u'(a) = \alpha\, u(a).$$

®

The statement u' (a) = α u (a) gives us the admissible transformation for a ratio scale.

Modified Extensive Structure

Definition 4.x: (Modified Extensive Structure):

Let **P** be a non-empty set, •≥ binary relation on A, and o a closed binary operation on **P**. The relational system (**P**, •≥, o) is an **extensive structure** if and only if the following axioms hold for all P1,..,P4 ∈ **P**.

A1': (**P**,•≥) is a weak order

A2': P1 o (P2 o P3) ≈ (P1 o P2) o P3,
axiom of weak associativity

A3': P1 o P2 ≈ P2 o P1,
axiom of weak commutativity

A4': P1 •≥ P2 => P1 o P3 •≥ P2 o P3
axiom of weak monotonicity

A5': If P1 •> P2 then for any P3,P4 there exists a natural number n, such that nP1 o P3 •> nP2 o P4,
Archimedian Axiom
®

Both extensive structuresareequivalent.

Concatenation Operations

Definition (Binary Operation**):**
A binary operation is a mapping of: A x A into A
®

Important is that holds:

A x A -> A. Every combination of (a∈ A) x (b∈ A) and

a o b

maps into the set A, again.

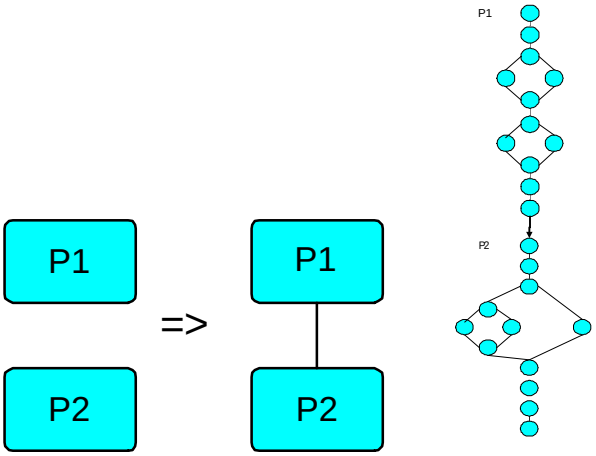


Figure 11: Concatenation Operation BSEQ.

Binary Operation

Definition (Binary Operation**)**

A x A into A

®

Concatenation Table

Assume, we have a finite set of Flowgraphs F1-F5, F11-F15, F21-F25, F31-F35, F41-F45, and F51 to F55 P.

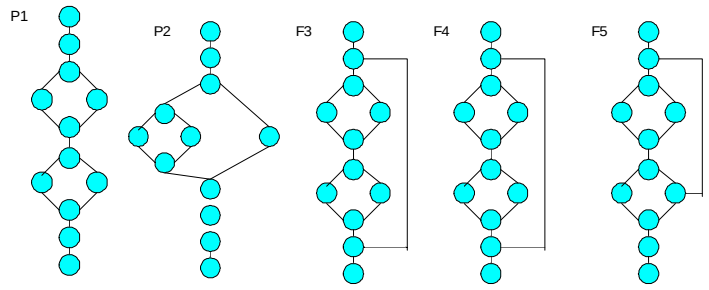


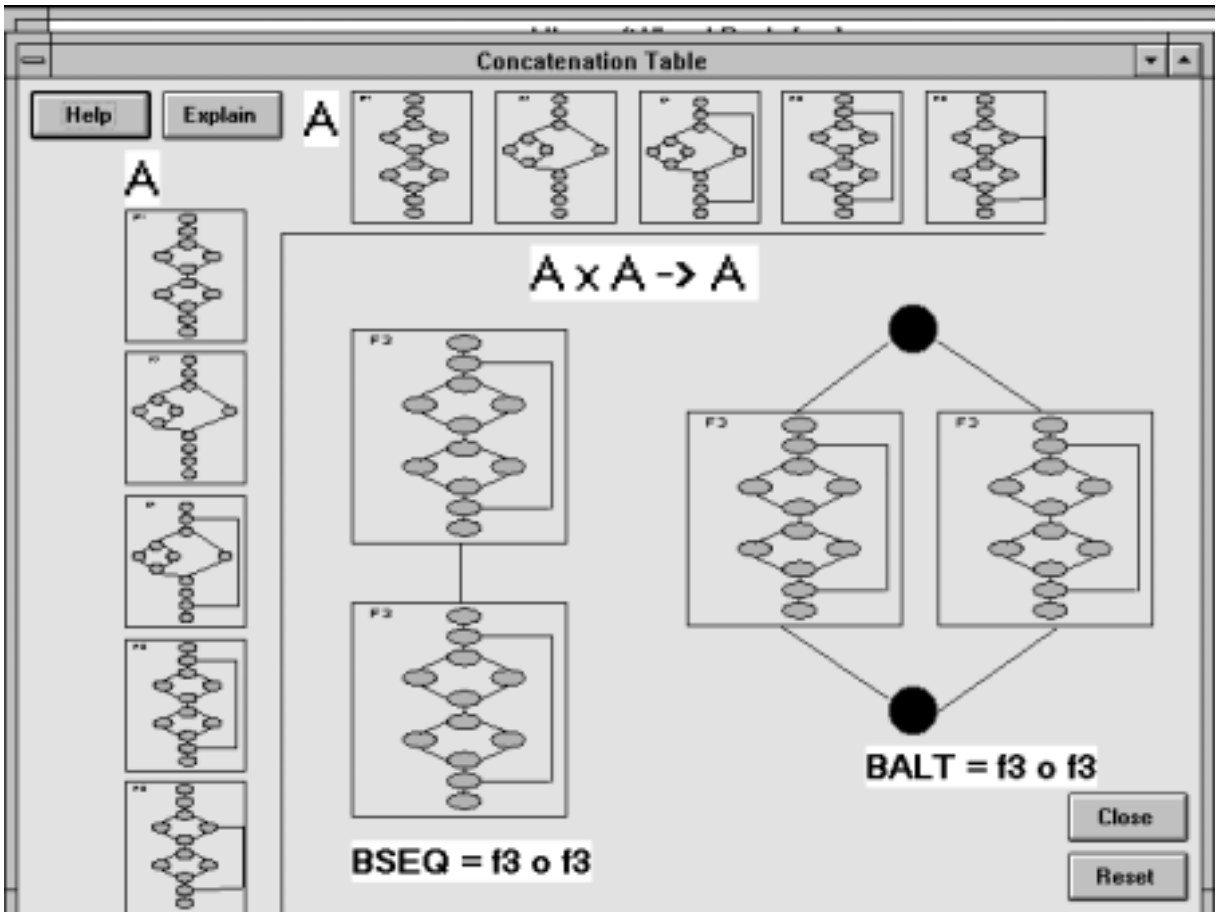
Figure 12: Flowgraphs F1-F5.

In this table we have the Flowgraphs F1-F5 in a row and a column in an arbitrary sequence. We denote the concatenation of F1 with F1 as F1oF1 and use the abbreviation F11.

	F1	F2	F3	F4	F5
F1	F11	F12	F13	F14	F15
F2	F21	F22	F23	F24	F25
F3	F31	F32	F33	F34	F35
F4	F41	F42	F43	F44	F45
F5	F51	F52	F53	F54	F55

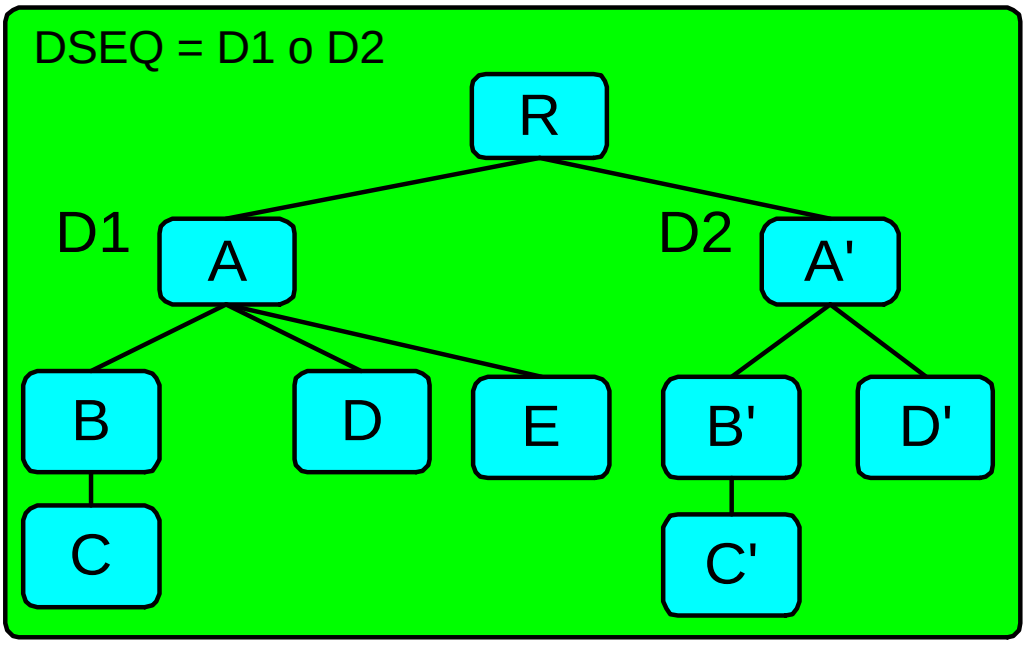
Figure 13: Concatenation table with the finite set F1 - F5..

Concatenation Operations



Concatenation Operations

Structure Chart



Concatenation Operations

OO-Environment

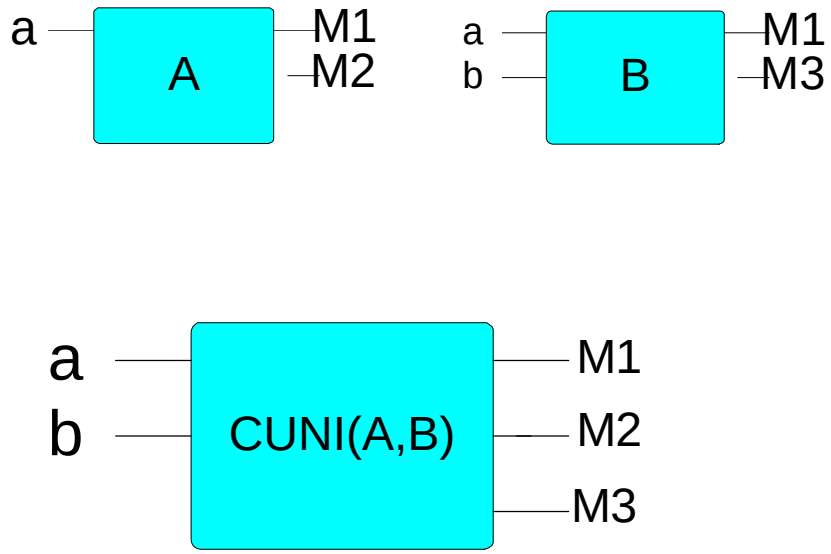


Figure 14: Concatenation Operation CUNI(A,B).

Concatenation Operations

OO-Environment

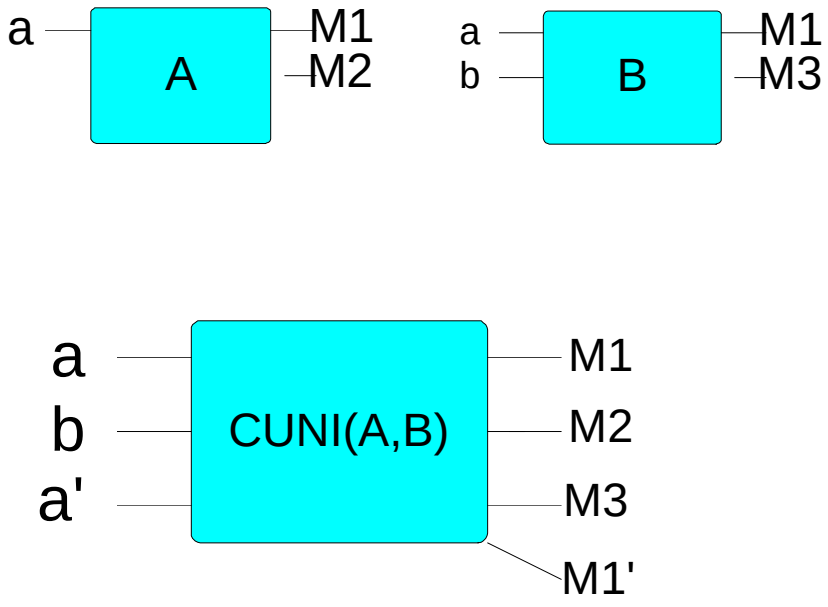


Figure 15: Again, the Concatenation Operation CUNI(A,B).

Concatenation Operations

OO-Environment

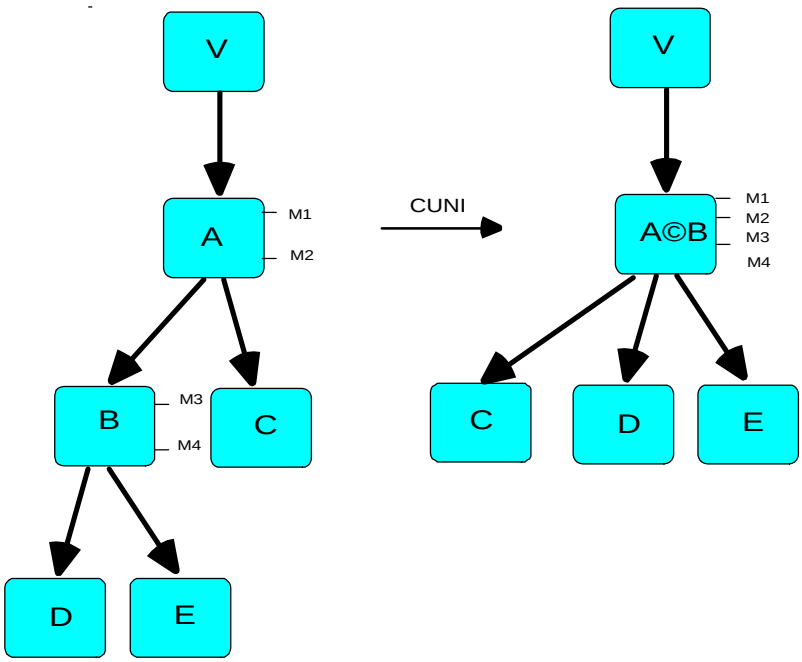


Figure 16: Concatenation operation CUNI(A,B).

Empirical Conditions of the Extensive Structure

Weak Order

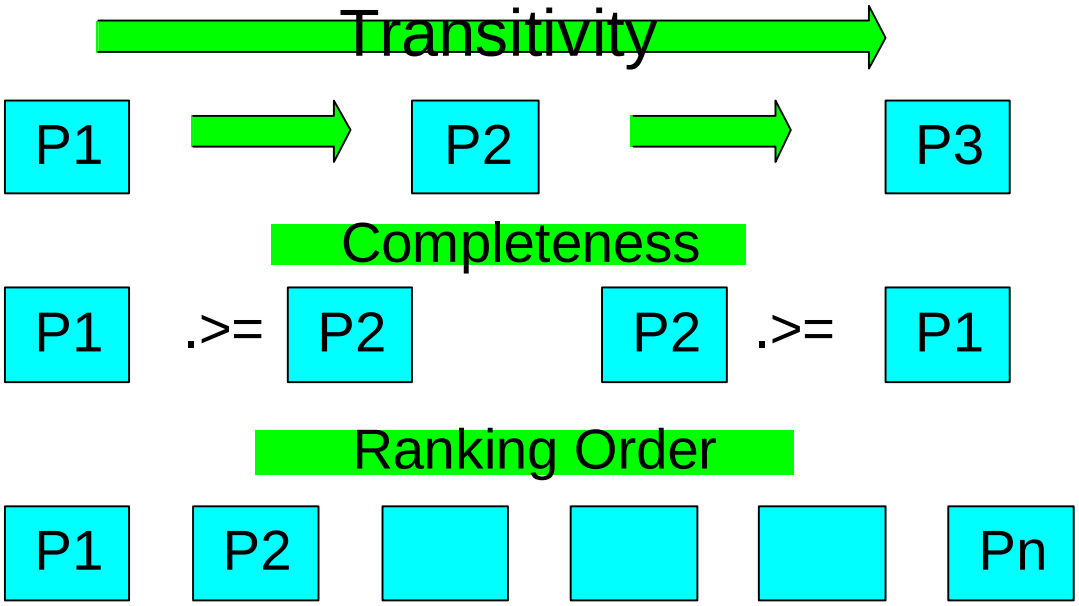


Figure 17: Weak order with transitivity and completeness. The weak order leads to the ordinal scale.

Empirical Conditions of the Extensive Structure

Weak Order / Ordinal Scale

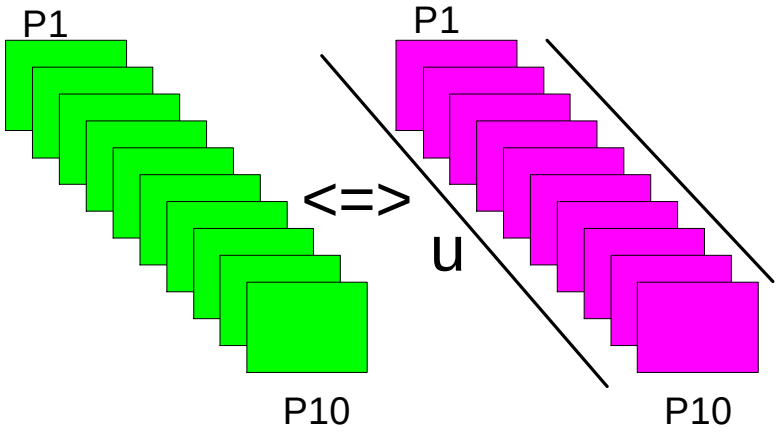


Figure 18: We have an ordinal scale if the empirical ranking order corresponds with the numerical ranking order.Weak order.

Empirical Conditions of the Extensive Structure

Weak Positivity

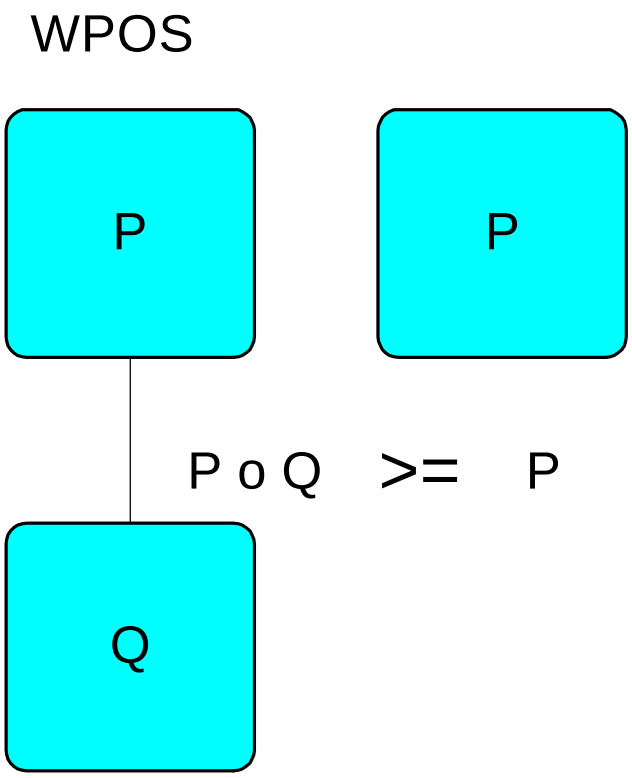


Figure 19: Axiom of weak positivity.

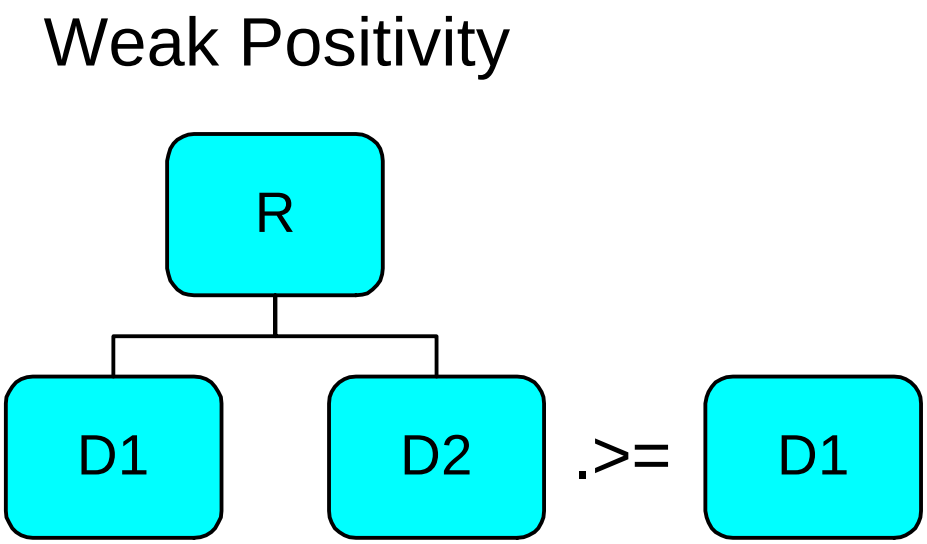


Figure 20: Axiom of weak positivity.

Empirical Conditions of the Extensive Structure

Weak Associativity

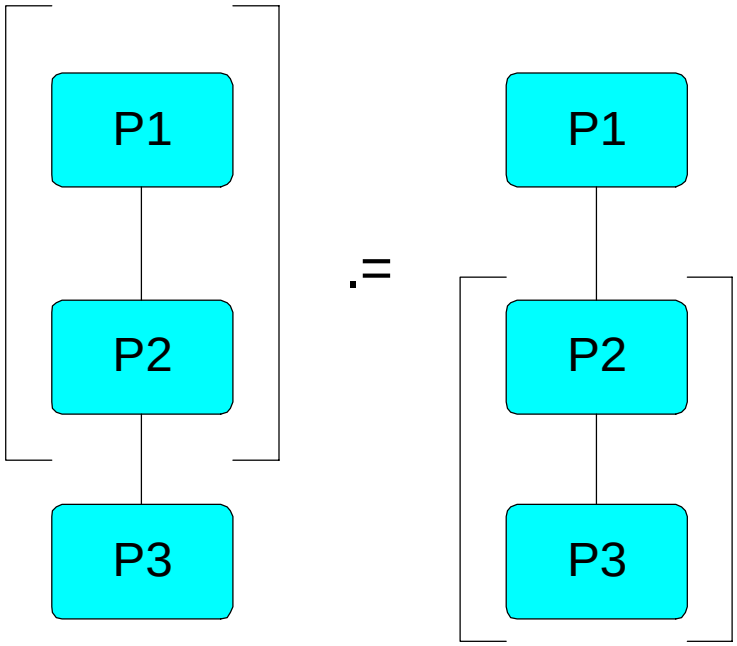


Figure 21: Axiom of weak associativity.

Empirical Conditions of the Extensive Structure

Weak Associativity for Structure Charts

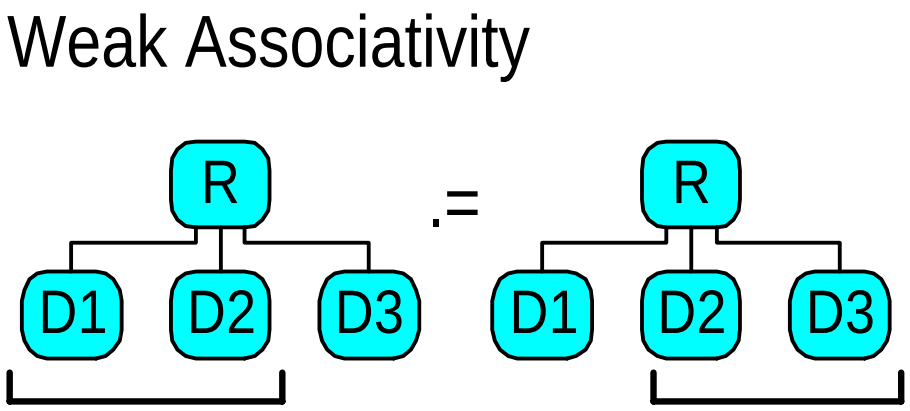


Figure 22: Axiom of weak associativity.

Empirical Conditions of the Extensive Structure

Weak Associativity

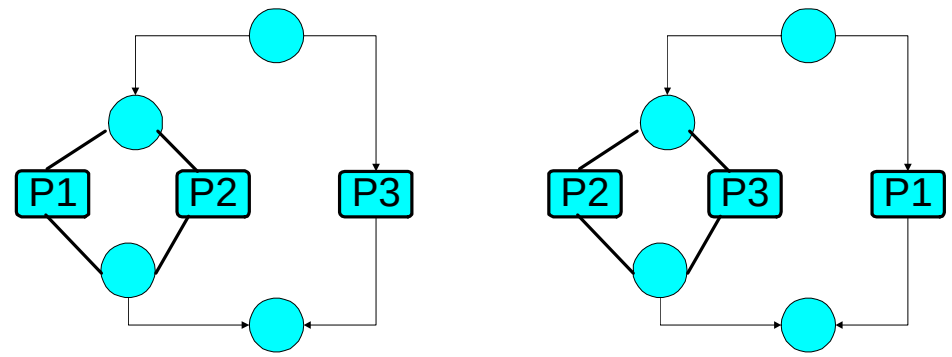


Figure 23: Axiom of weak associativity for the Concatenation Operation BALT..

Empirical Conditions of the Extensive Structure

Weak Commutativity

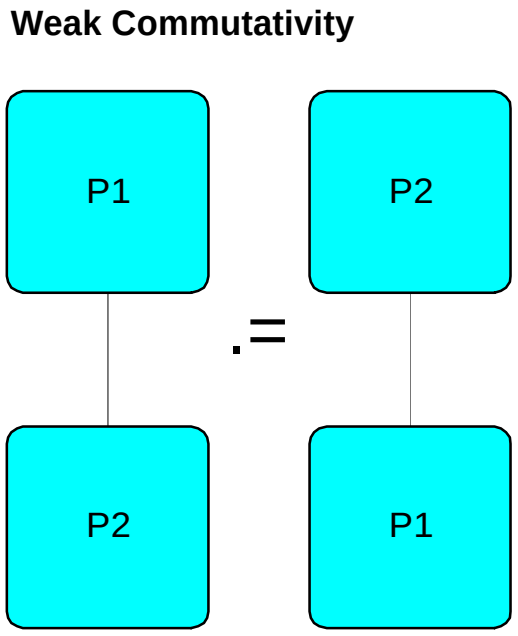


Figure 24: Axiom of weak commutativity.

Empirical Conditions of the Extensive Structure

Weak Commutativity for Structure Charts

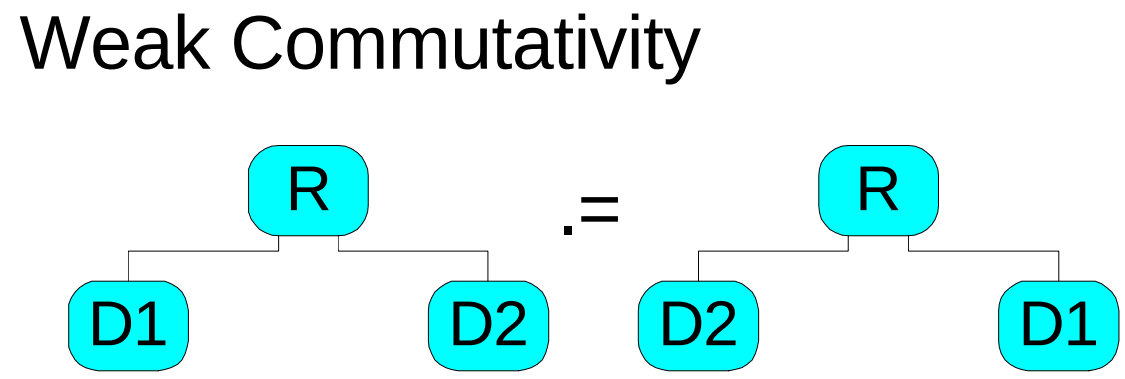


Figure 25: Axiom of weak commutativity.

Empirical Conditions of the Extensive Structure

Weak Monotonicity

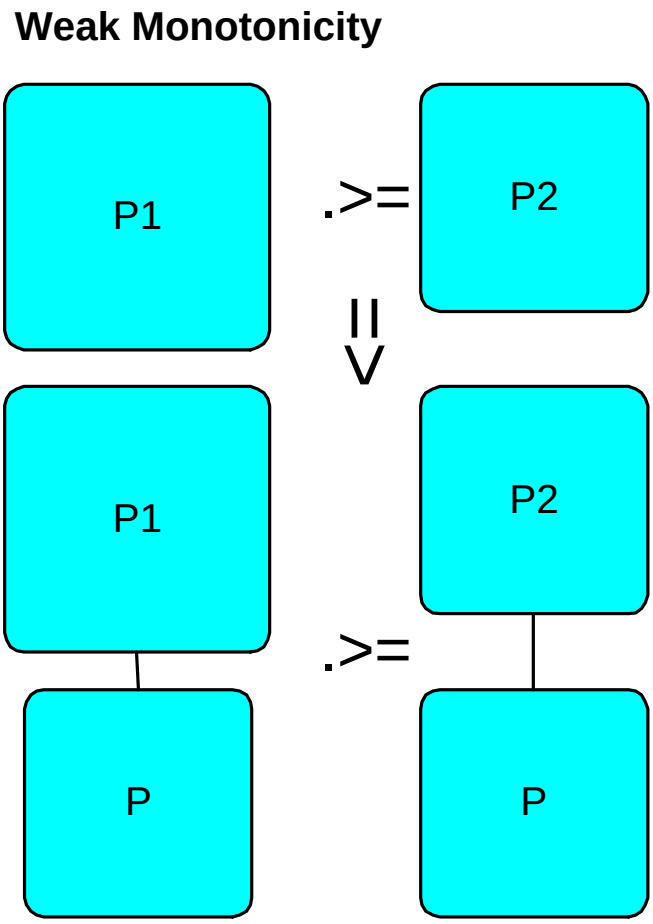
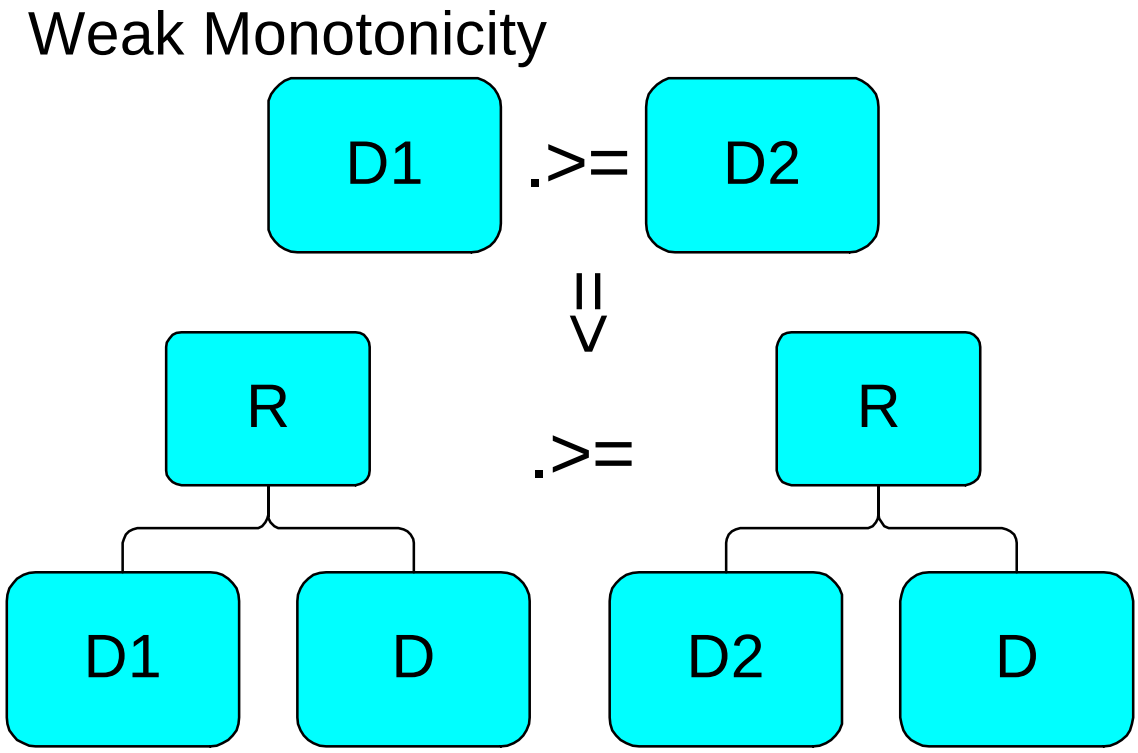


Figure 27: Axiom of weak monotonicity.

Empirical Conditions of the Extensive Structure

Weak Monotonicity for Structure Charts



Substitution Property

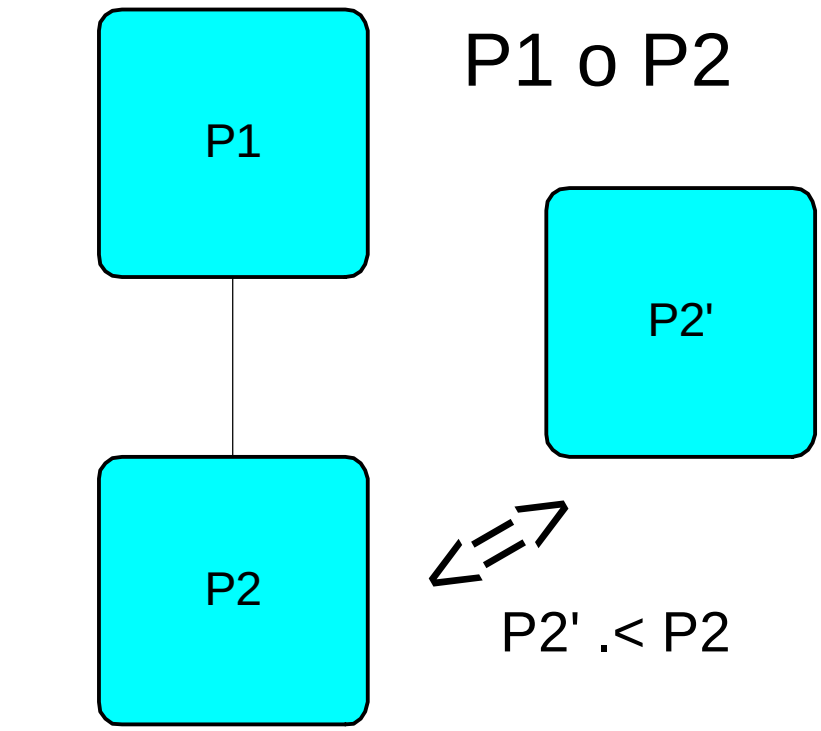


Figure 28: Axiom of weak monotonicity - substitution property.

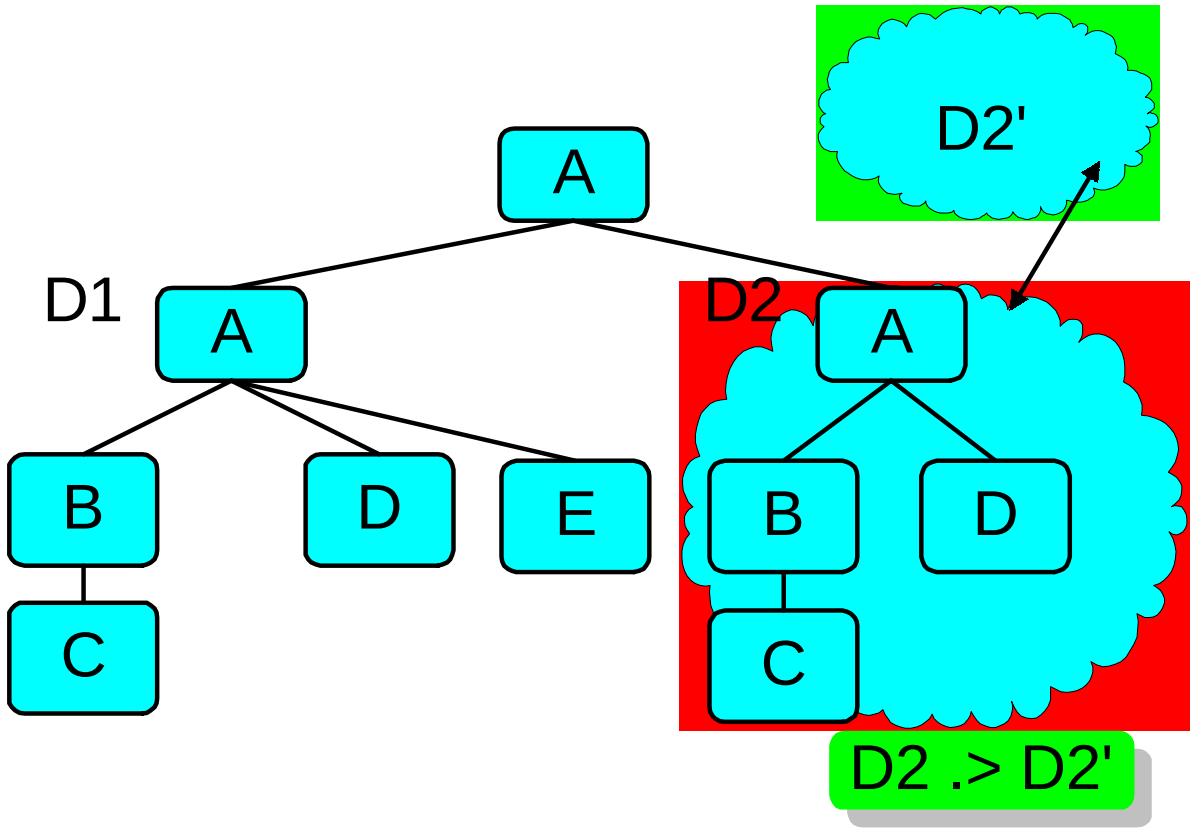


Figure 29: Axiom of weak monotonicity - substitution property.

Empirical Conditions of the Extensive Structure

Archimedian Axiom

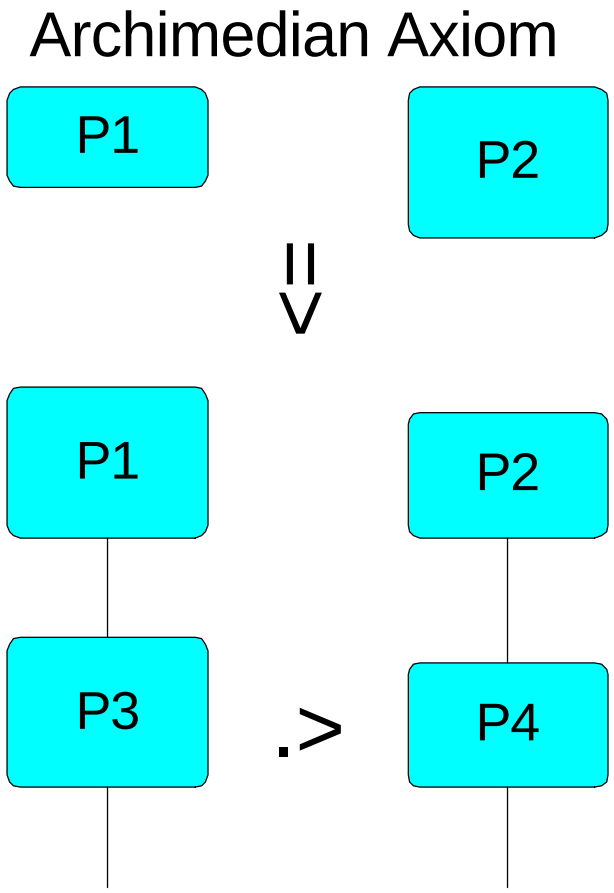


Figure 30: Archimedian Axiom.

Empirical Conditions of the Extensive Structure

Archimedian Axiom for Structure Charts

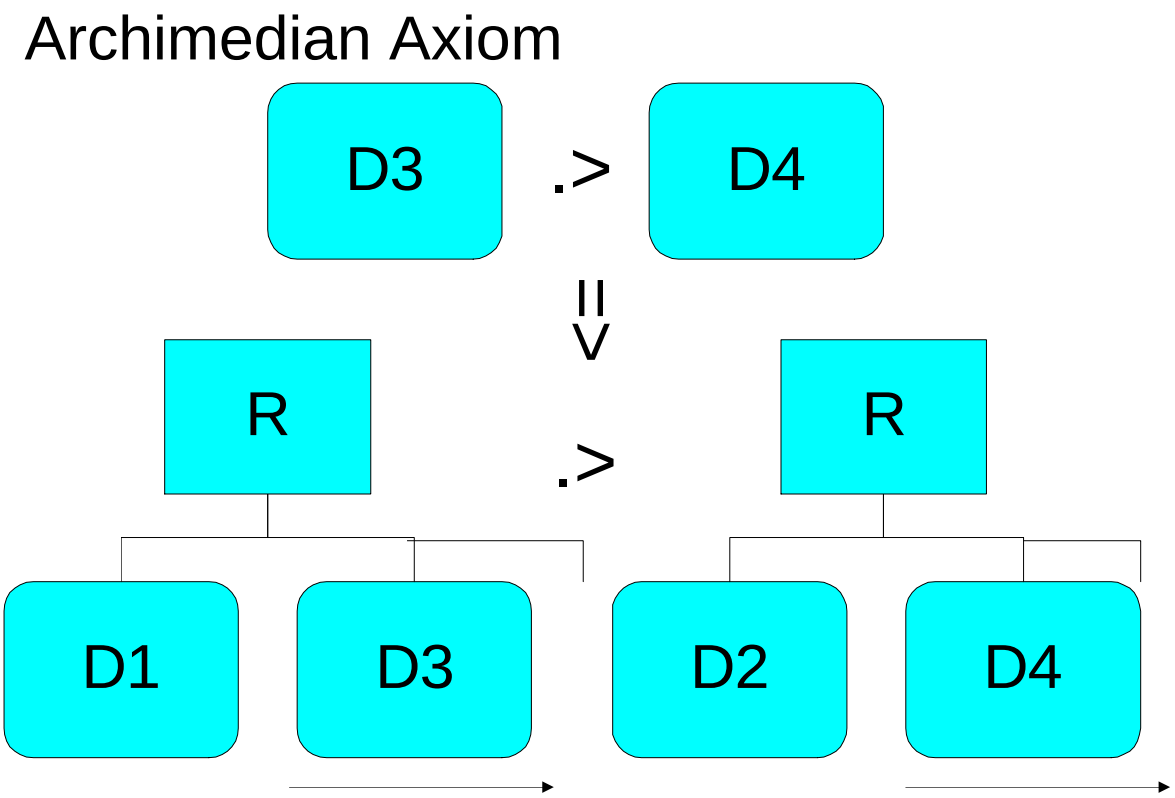


Figure 31: Archimedian Axiom.

Extensive Structure

Every additive measure assumes an extensive structure.

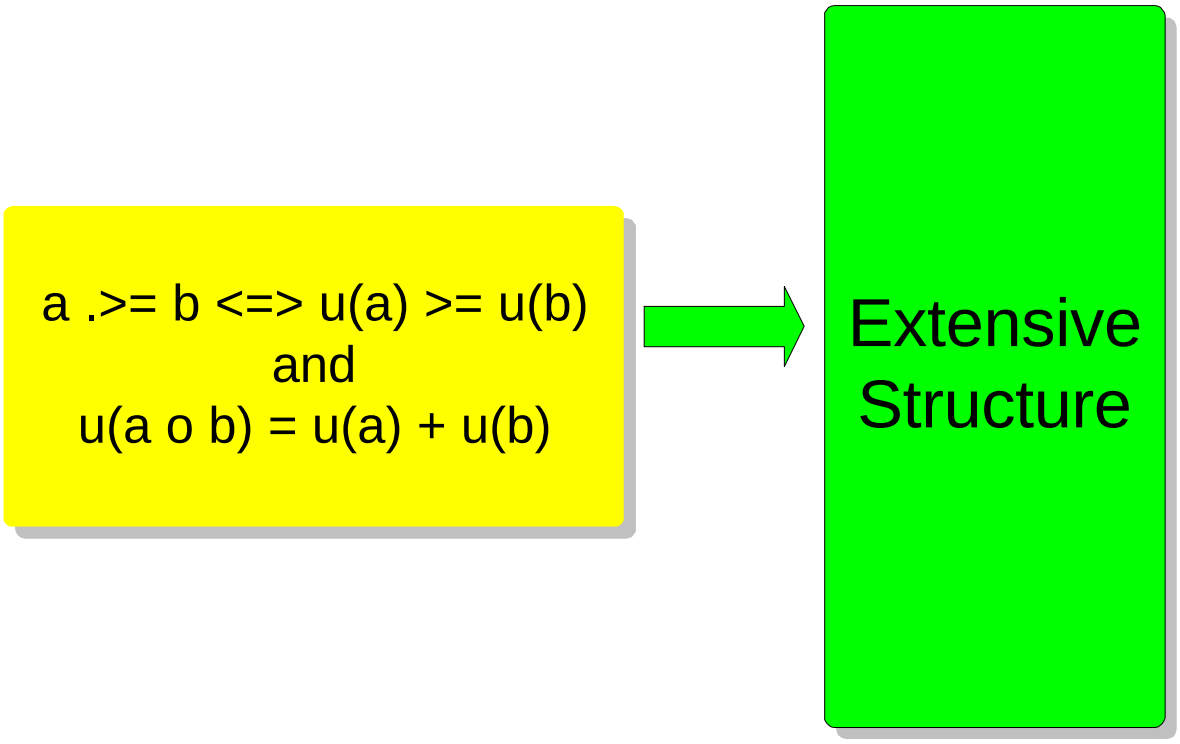


Figure 32: Every measure which is additive assumes an extensive structure.

Extensive Structure for Flowgraphs

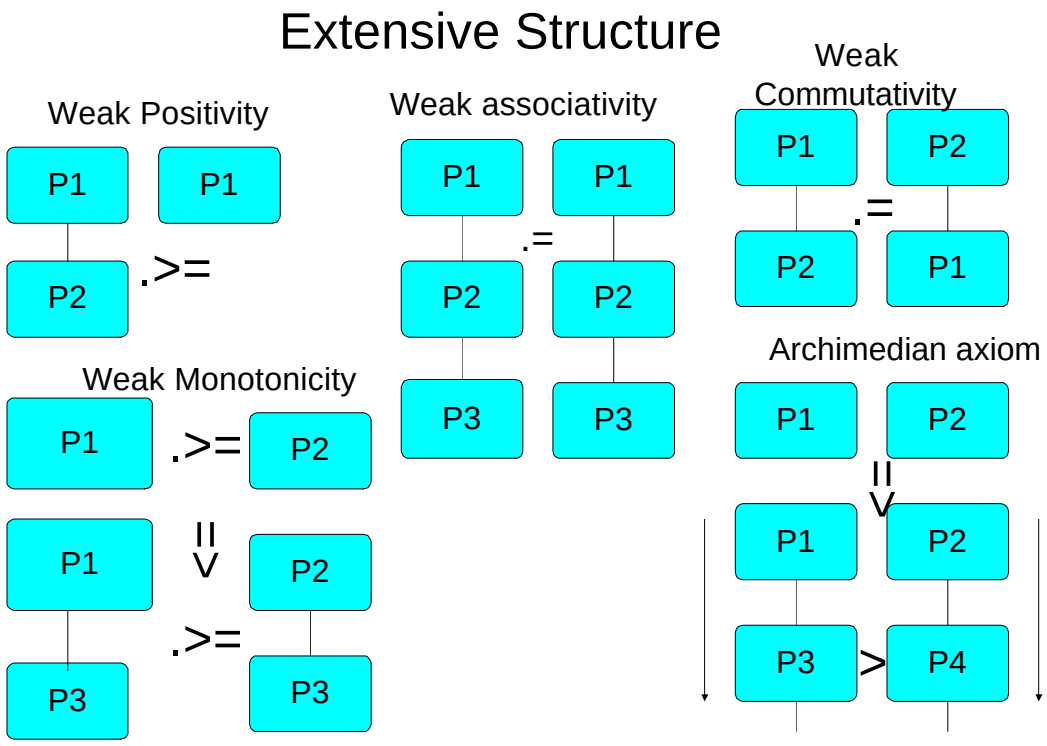


Figure 33: Extensive structure for flowgraphs.

Extensive Structure for Structure Charts

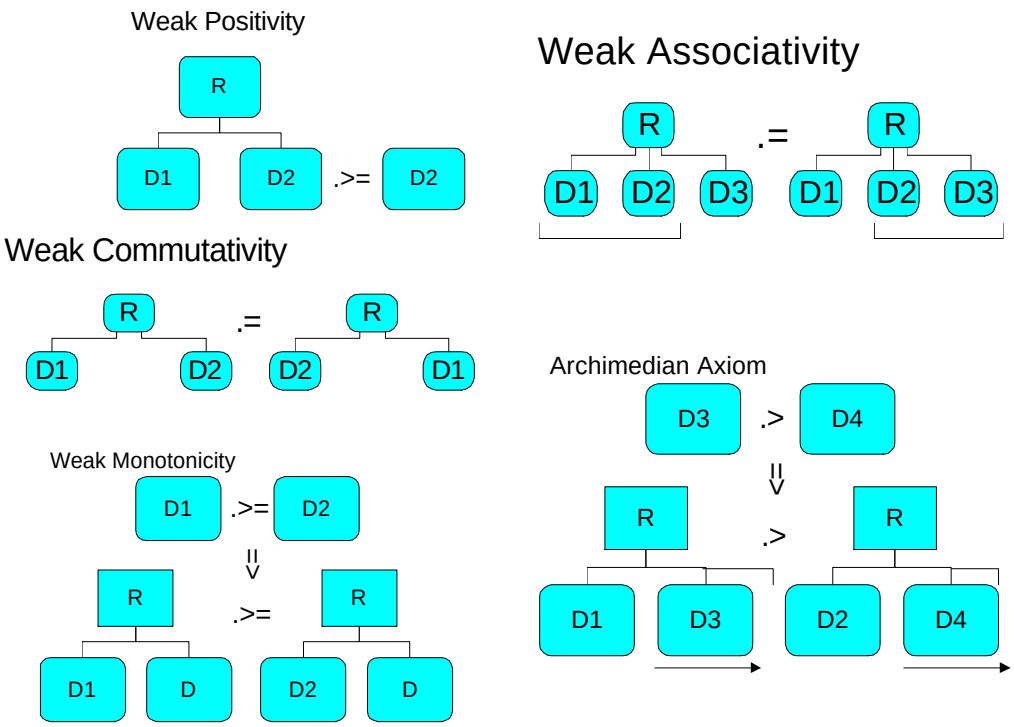


Figure 34: Extensive structure for structure charts.

Extensive Structure in the Object-Oriented Area

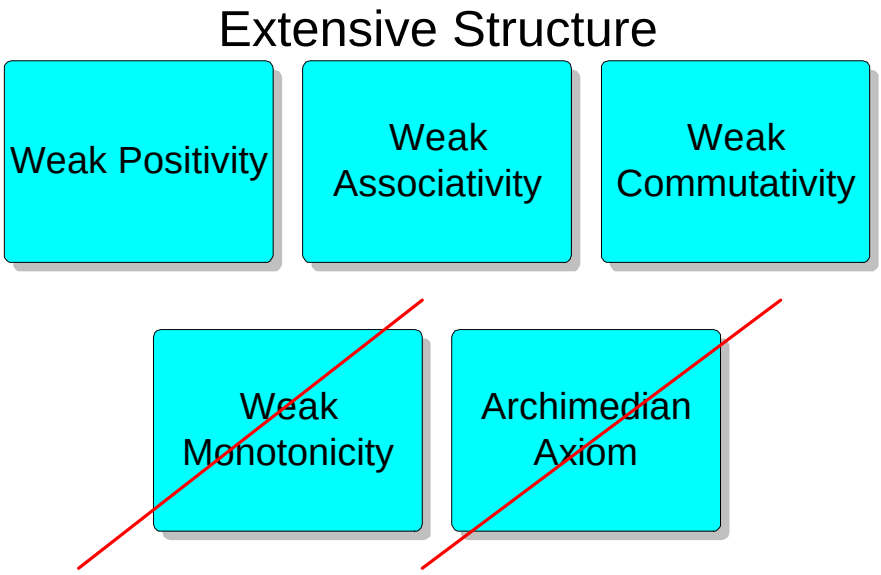


Figure 35: Object-oriented measures mostly do not assume an extensive structure.

Discussion of the Extensive Structure

1. We say, a measure assumes an extensive structure or it does not assume an extensive structure.. This includes that a measure assumes the weak order, the axiom of weak positivity, etc., or not.

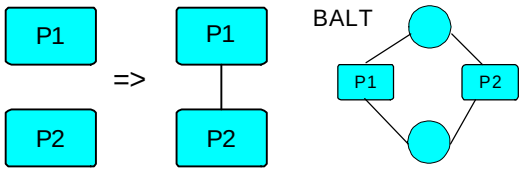
2. We cannot say that a Measure u is a ratio scale. For a ratio scale we need a homomorphism. Saying, a measure assumes an extensive structure assumes a certain ranking order of objects and an empirical binary / concatenation operation. We say, a measure can be used as an ordinal, interval, ratio or absolute scale. The term *can be* means here the case of a homomorphism.

3. We can say that the Measure LOC can be used as an ordinal scale, or the Measure of McCabe can be used as an ordinal scale. In this case both measures assume a weak order, but the both weak orders are differently.

4. Not every measure, which assumes an extensive structure can be used as a ratio scale. Examples are strictly monotonic transformations of an additive measure. However, under certain conditions non-additive ratio scales are possible.

Measures can assume an extensive structure, although the Measure u is not additive.

Measures and Extensive Structure



Measures of McCabe 1976

MCC-V = |E| - |N| + 2:

- Assumes extensive structure
- Non-additive combination rule
- Combination rule: u(P1 o P2) = u(p1) + u(P2) -1
- Combination rule is not meaningful for the ratio scale

MCC-V2 = |E| - |N| + 1:

- Assumes extensive structure
- Additive combination rule
- Combination rule: u(P1 o P2) = u(p1) + u(P2)
- Combination is rule meaningful for the ratio scale
- Additive Ratio Scale

Strong Monotonic Functions

MCC-V2 = MCC-V - 1

or

MCC-V = MCC-V2 + 1

Independence Conditions

The independence conditions consider the following question:

$$u(P1 \circ P2) = f(u(P1), u(P2))?$$

The question is whether a function f exists, what the function f areand what the weakest condition for the existence of f is.

The answer is given by the independence conditions C1 - C4.

C1:

$a \approx b \Rightarrow a \circ c \approx b \circ c$, and $a \approx b \Rightarrow c \circ a \approx c \circ b$.

C2:

$a \approx b \Leftrightarrow a \circ c \approx b \circ c$, and $a \approx b \Rightarrow c \circ a \approx c \circ b$.

C3:

$a \bullet\geq b \Rightarrow a \circ c \bullet\geq b \circ c$, and $a \bullet\geq b \Rightarrow c \circ a \bullet\geq c \circ b$.

C4:

$a \bullet\geq b \Leftrightarrow a \circ c \bullet\geq b \circ c$, and $a \bullet\geq b \Rightarrow c \circ a \bullet\geq c \circ b$.

The weakest condition for the existence of f is C1.

Independence Condition C1 and C2

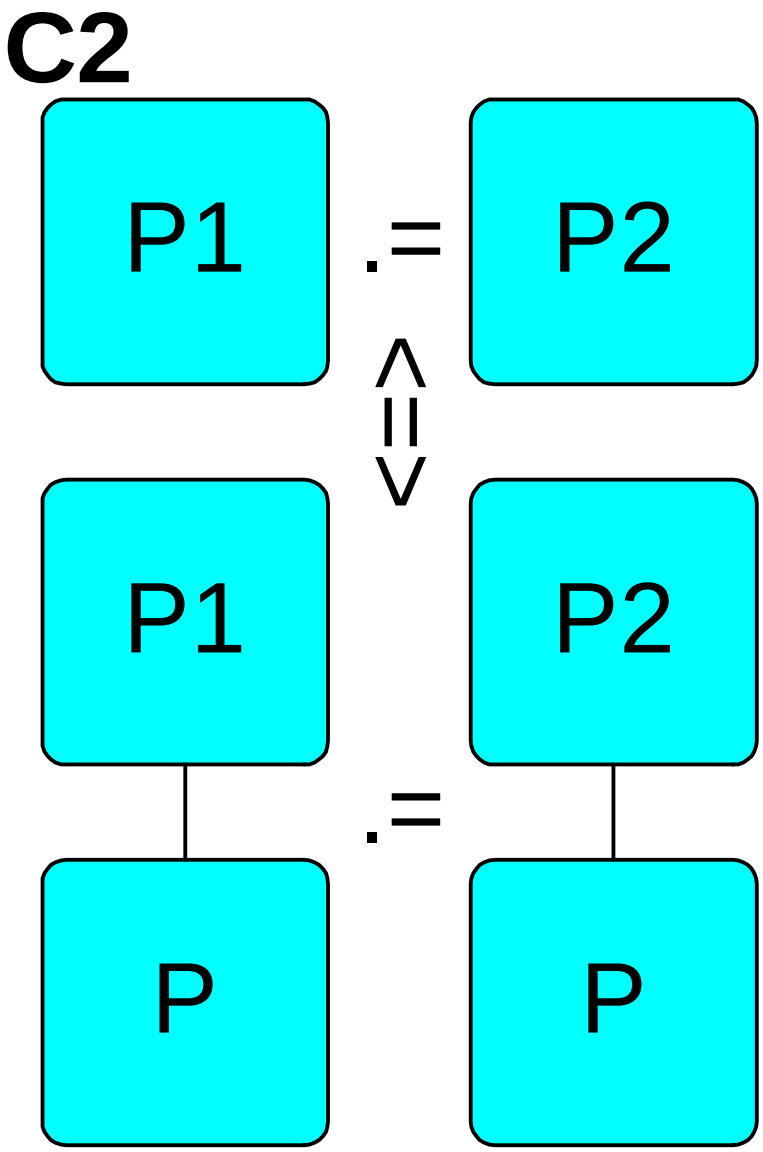


Figure 36: Independence Conditions C1-C2. For C1 holds => and for C2 holds <=>.

Independence Condition C3 and C4

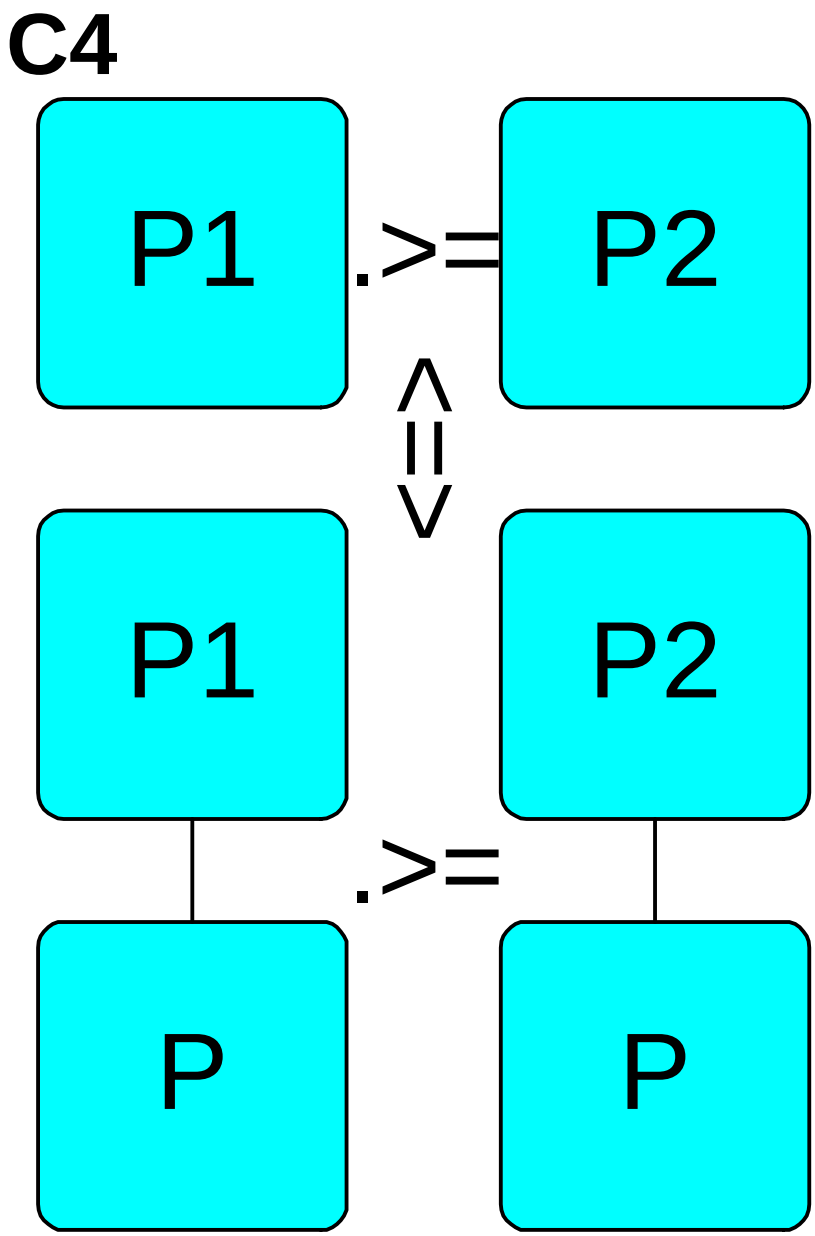


Figure 37: Independence Conditions C3-C4. For C3 holds => and for C4 holds <=>.

Hierarchy of Independece Conditions

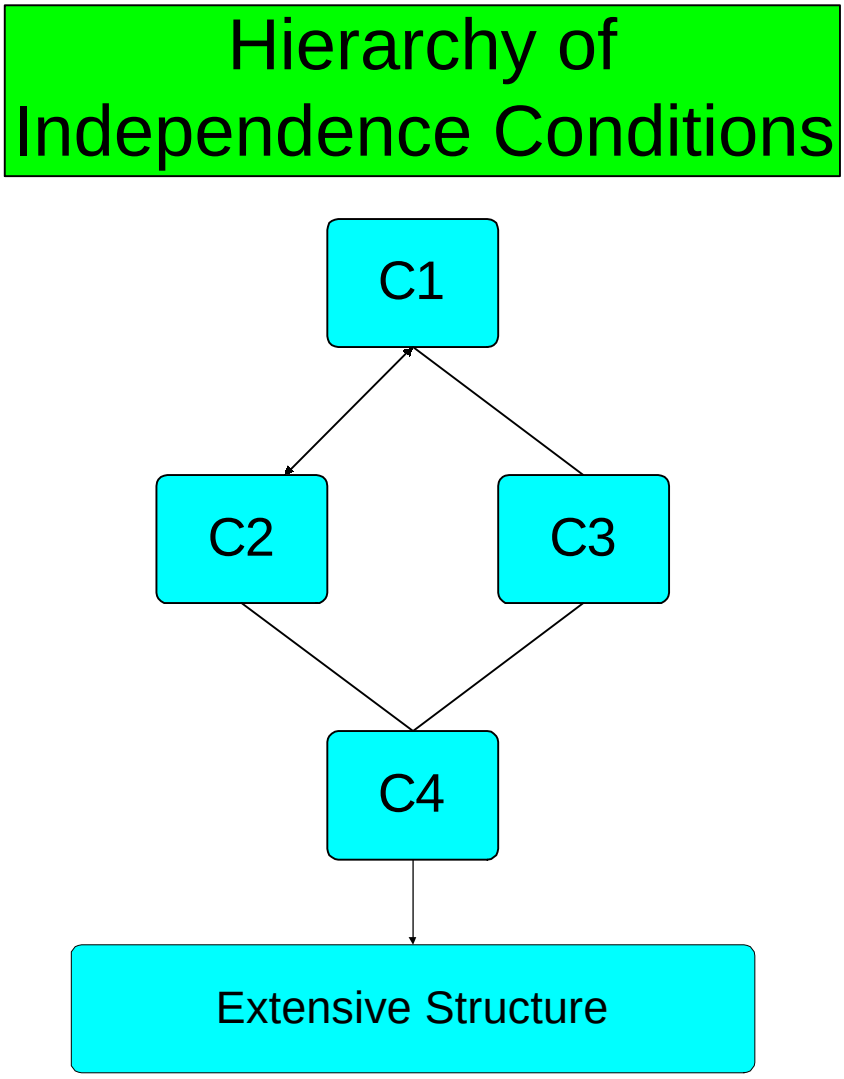


Figure 38: Hierarchy of Independence Conditions C1-C4.

Measure Defect-Density

DD = Defects / LOC

VERSION1				VERSION2			
#	DEFECT1	LOC1	D1/LOC1	DEFECT2	LOC2	D2/LOC2	REL-1-2
1	12	777	0.01544	3	55	0.05455	<
2	5	110	0.04545	6	110	0.05455	<
3	2	110	0.01818	3	110	0.02727	<
4	3	110	0.02727	4	110	0.03636	<
5	6	1000	0.00600	70	10000	0.00700	<
6	28	2107	0.01329	86	10385	0.00828	>

.

Figure 39: Consequences of violating the independence condition C1 which implies a violation of the axiom of weak monotonicity.

Substitution Property of the Measure Defect-Density

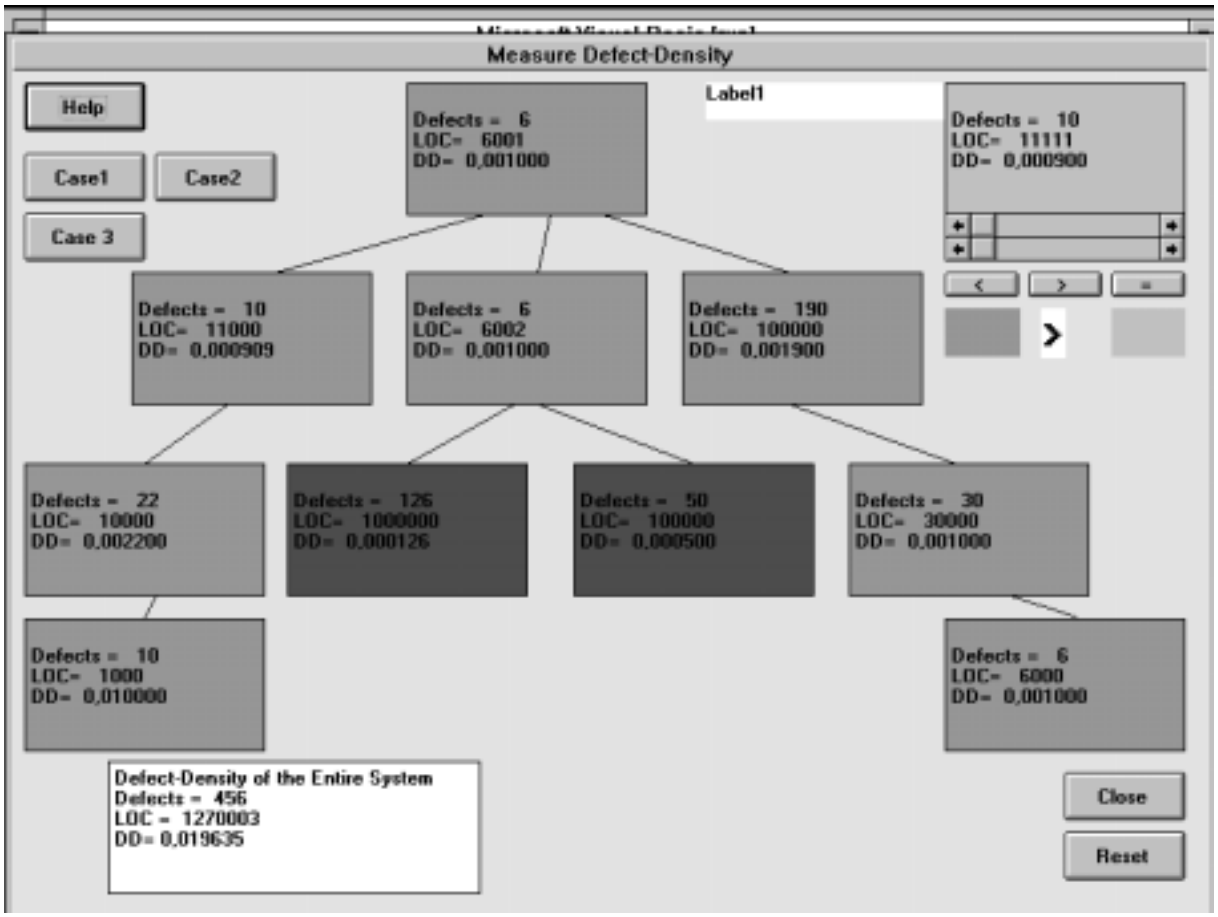


Figure 40: A hierarchical ordered software system. each module has a certain Defect-Density.

Substitution Property of the Measure Defect-Density

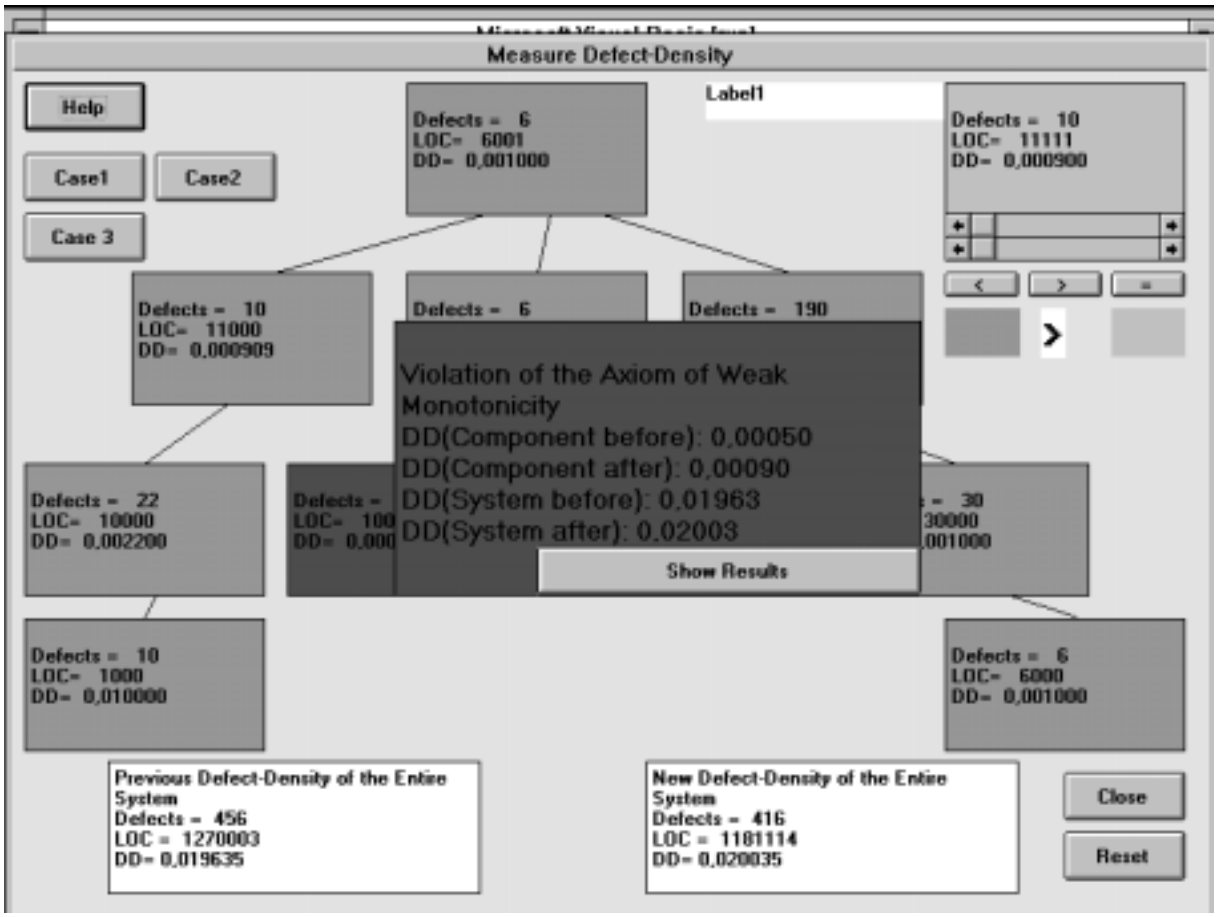


Figure 41: A hierarchical ordered software system. each module has a certain Defect-Density. A green module is replaced by a grey one.

Modification of the Measure Defect-Density

Assume it holds in reality:

$$\textbf{Defects} = a\,LOC^{b},\ a>0,\ b>1.$$

with a>0, b> 1.

It follows by insertion:

$$DD' = a\,LOC^{b-1}$$

with a>0, b>1.

Results:

1. DD does not assume the weakest independence condition C1

2. DD cannot be used as a ratio scale.

3. The question is whether holds in reality: Defects = a LOC^{b}, a>0, b>1.

4. If yes:

5. DD' is a proper measure:

6. DD assumes an extensive structure

7. DD assumes a non-additive ratio scale (see below).

3 Weyuker Conditions

In 1988 Weyuker proposed desirable properties for software measures. We discuss some of them.

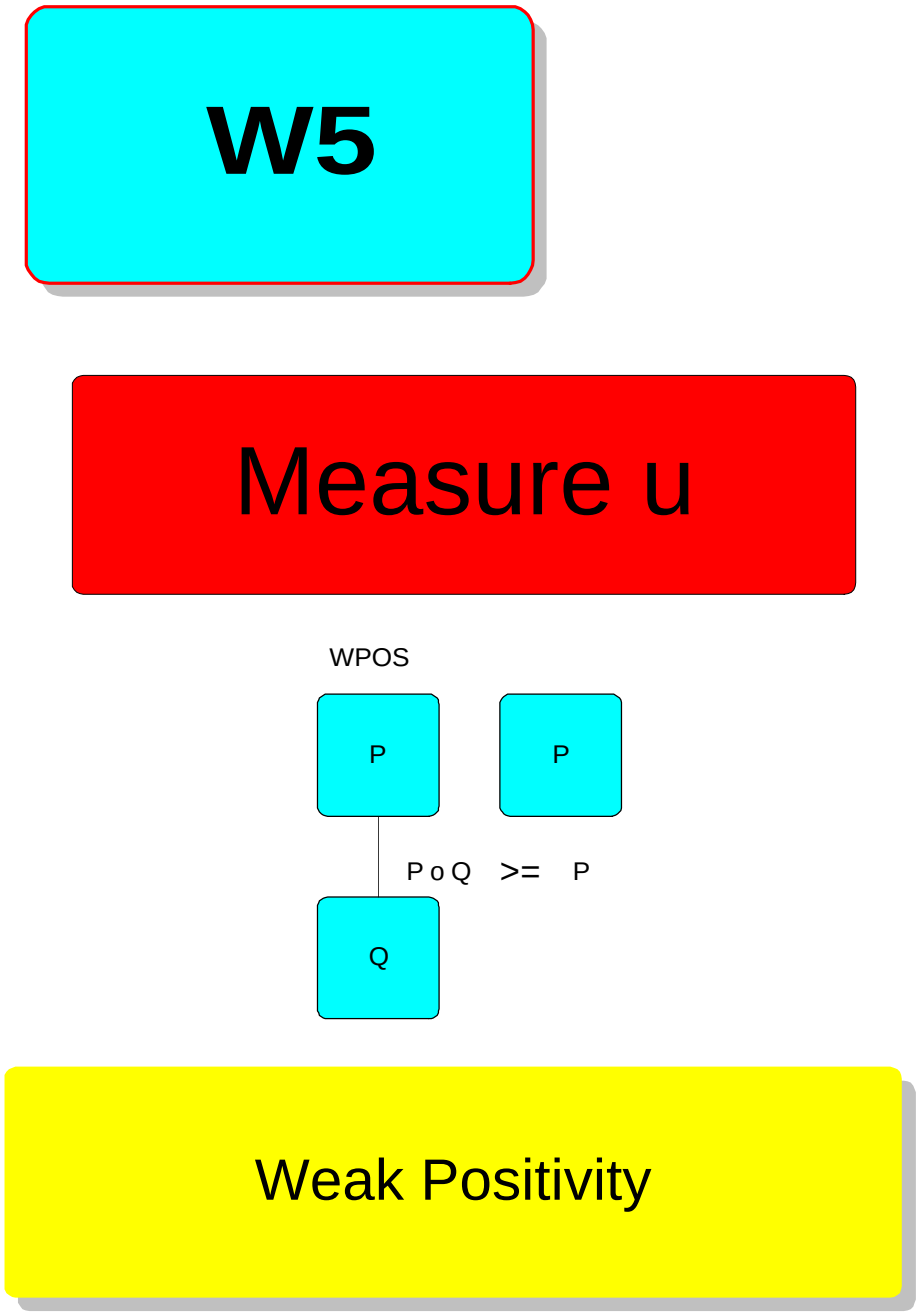


Figure 42: Weak positivity is an axiom of the (positive) extensive structure.

Weyuker Conditions (cont.)

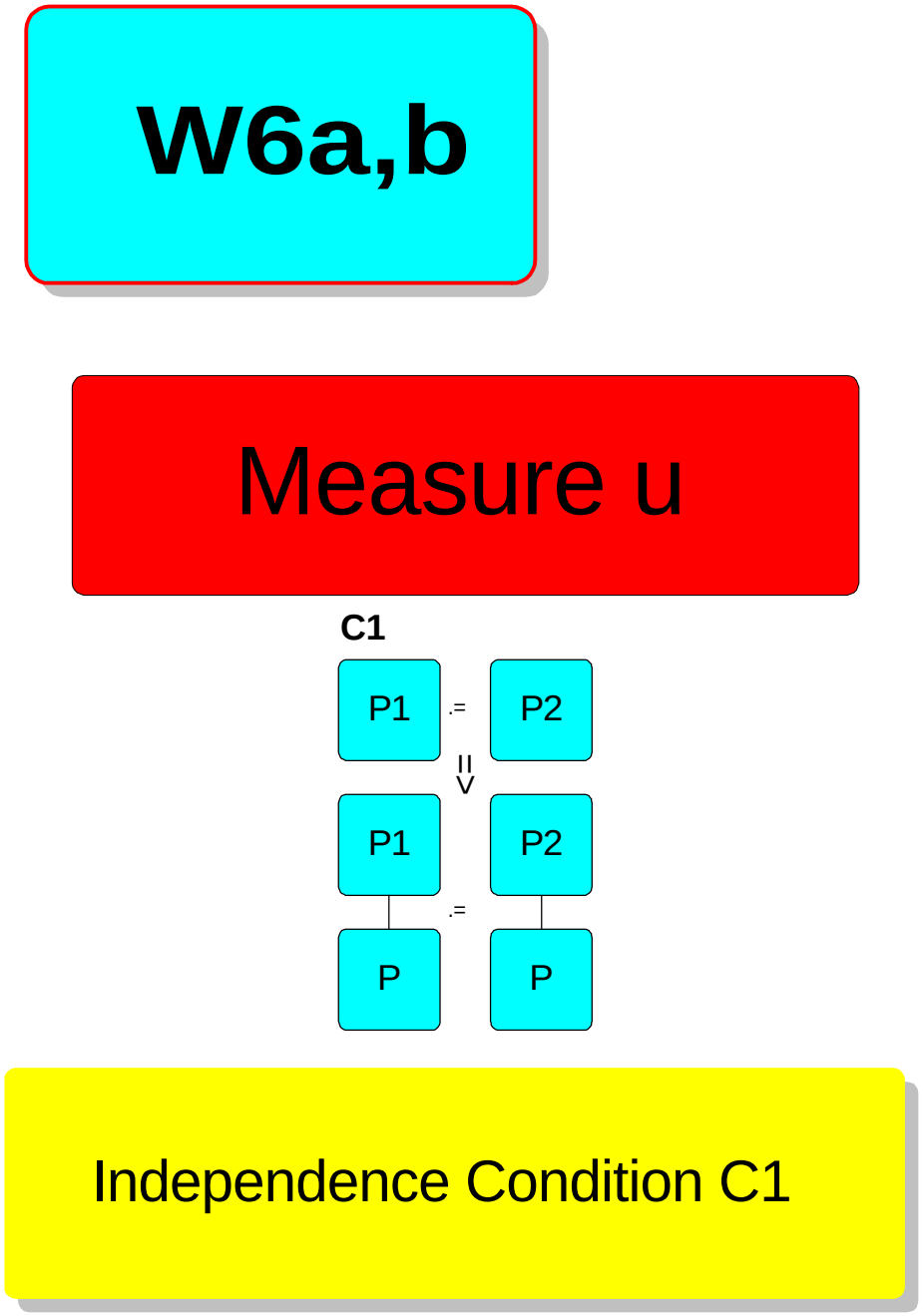
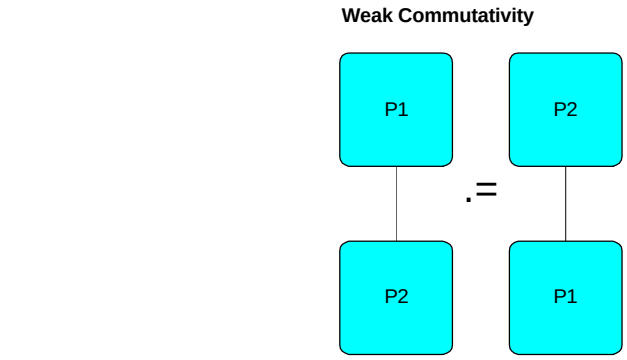


Figure 43: The Independence Condition C1 is assumed by the extensive structure.

Weyuker Conditions (cont.)

W7

Measure u



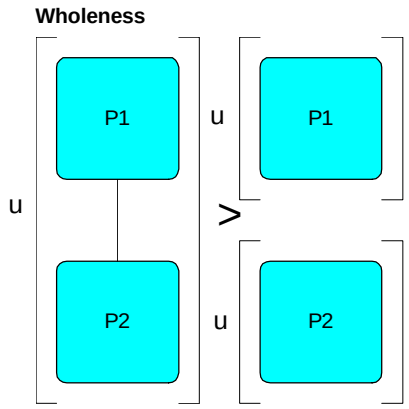
Weak Commutativity

Figure 44: The axiom of weak commutativity is an axiom of the extensive structure.

Weyuker Conditions (cont.)

W9

Measure u



Wholeness
u(P1 o P2) > u(P1) + u(P2)

Figure 45: Wholeness requires a ratio scale.

Wholeness requires a ratio scale because the statement above is meaningful for a ratio scale.

Weyuker Conditions (cont.)

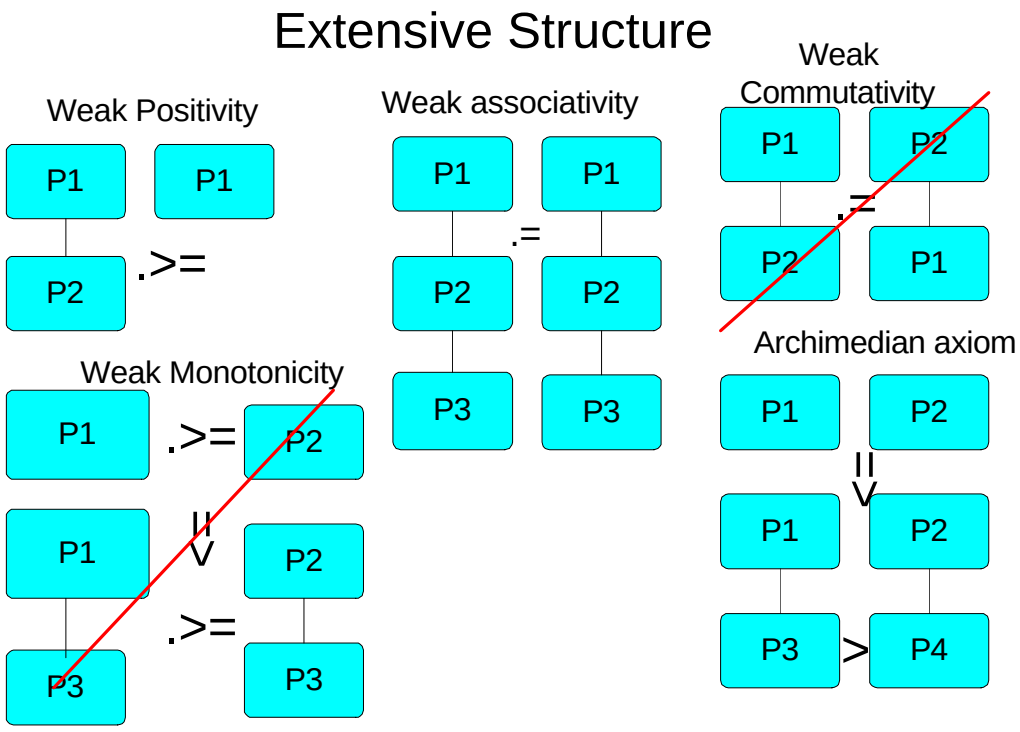


Figure 46: Wholeness requires a non-additive ratio scale.

The question is what is a non-additive ratio scale.

Wholeness

$$u\,(P1 \circ P2) > u(P1) + u(P2).$$

Admissible transformation of the ratio scale: $g(x) = ax$, $a>0$. It holds:

$$au\,(P1 \circ P2) > au(P1) + au(P2).$$

This statement is meaningful for a ratio scale.

We remember:

$$u\,(P1 \circ P2) = u(P1) + u(P2)$$

is an additive combination rule. It is also meaningful for the ratio scale. It assumes an extensive structure.

Non-Additive Ratio Scale

In order to get a non-additive ratio scale, we have to look for a non-additive combination rule, which is meaningful for the admissible transformation of the ratio scale.

Such a combination rule has the following structure:

$$u(P1 \circ P2) = (u(P1)^{1/b} + u(P2)^{1/b})^{b}.$$

It holds:

$b = 1$	Additive combination rule -> extensive structure
$b > 1$	Supra-additivity - Wholeness, non-additive combination rule -> extensive structure
$b < 1$	Sub-additivity, non-additive combination rule -> extensive structure

How does a Measure look like for a Non-additive Combination Rule?

Measure u' = u^b, b $\in$ $\Re$>0.:

1. Measure u' has a non-additive combination rule

2. Measure u' has a combination rule, which is meaningful for the ratio scale.

3. Measure u' can be used as a non-additive ratio scale measure.

Wholeness - Additive and Non-additive Ratio Scale

It holds:

1. Wholness requires a non-additive ratio scale.

2. If b=1, then we have an additive ratio scale.

3. The modification of b does not change the empirical relational system, it is only a modification of the numerical relational system.

4. Wholeness does not include new empirical properties of the measure, it is a pseudo-property without any empirical meaning.

5. It can be seen as a calibration of a measure.

Because wholeness requires an extensive structure, the Weyuker conditions are not compatible.

Demonstration of Wholeness with the Measure IF4

IF4 = Number of Fan-in in a module.

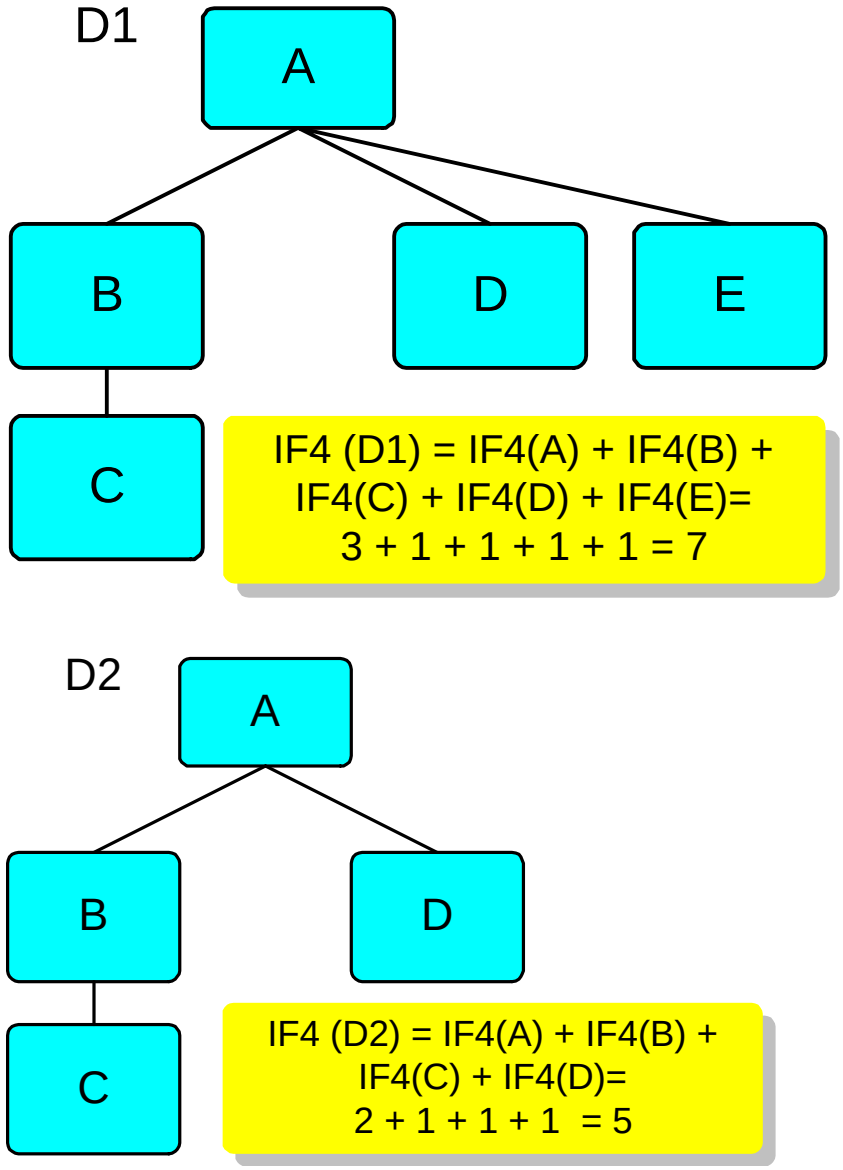
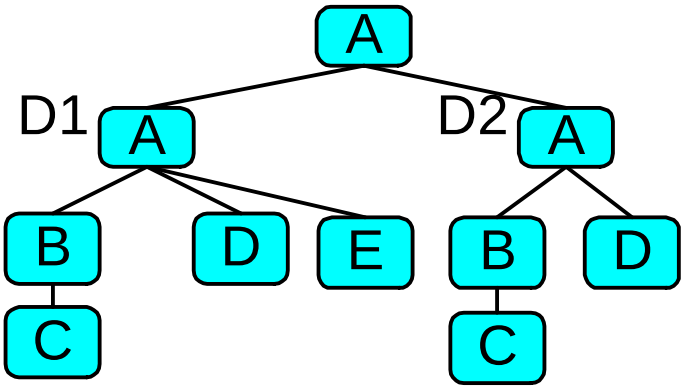


Figure 47: Measure IF4.

Measure IF4’ = IF4 + 4



IF4 (D1 o D2) = IF4(A) + IF4(B) + IF4(C) + IF4(D) + IF4(E) +
IF4(A') + IUF4(B') + IF4(C') + IF4(D')=
(3 + 1 + 1 + 1 + 1) + (2 + 1 + 1 + 1) + 4 = 17

Figure 48: Measure IF4.

IF4: non-additive, extensive structure, no ratio scale

IF4’ = IF4 + 4 => additive, extensive structure, additive ratio Scale

The Measures IF4 and IF’ assume the same extensive structure.

4 Validation of Software Measures and Prediction Models

- Many software measures are described in literature. Not only do these measures aim a wide range of attributes but also there are often many irreconcilable measures all claiming to measure or to predict the same attribute such as cost, size or complexity**.**

- The reason for this state of affairs is commonly attributed to a generally lack of validation of software measures.

- While accepting that reason, we propose more fundamentally that there is a lack of understanding of the meaning of validation of software measures and validation of software measures.

Two Separate Concepts of Validation

We should distinguish between two separate concepts of validation:

- Measures which are defined on certain objects and characterize numerically some specific attribute of these objects.

- Prediction systems involving a mathematical model and prediction procedures for it.

The fact, that people do not consider the differences between measures and prediction, has the consequence of the wrong interpretation of validation of software measures.

Definition 1 (Validation)

Validation of a software measure is the process ensuring that the measure is a proper numerical characterization of the claimed attribute.

Example 1

- A valid measure of the attribute of coupling of designs must not contradict to our intuitive notions about coupling.

- Specifically it requires a formal model for designs and a numerical mapping which preserves any relation over the set of designs which may intuitively imposed by the attribute of coupling.

- Thus the proposed measure of coupling should indeed measure precisely that; in particular if it is generally agreed that Design D1 has a greater level of coupling than Design D2, then any measure of coupling must satisfy

$$D1 \bullet\!> D2 \iff u(D1) > u(D2).$$

This is the basic concept of measurement and validation of a software measure.

Internal Validation

This type of validation is central.

- We can use the empirical relational system of measurement theory in order to validate a software measure.

- We also call this the **internal** validation of a software measure.

- Practitioners call this very often the ensuring of well-definedness and consistency of the measure.

Example

Internal Validation of the Measure of McCabe (Ranking Order)

MCC-V2 = E - N + 1

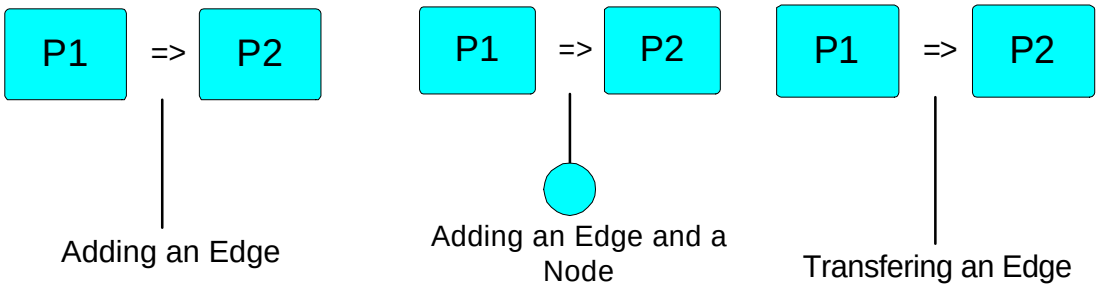


Figure 49: Internal Validation.

P1' results from P by adding an edge and a node.

Internal Validation of the Measure of McCabe (Additivity)

For the internal validation of a measure the characterization of the measure with

u(a o b) = u(a) + u(b)

is very helpful.

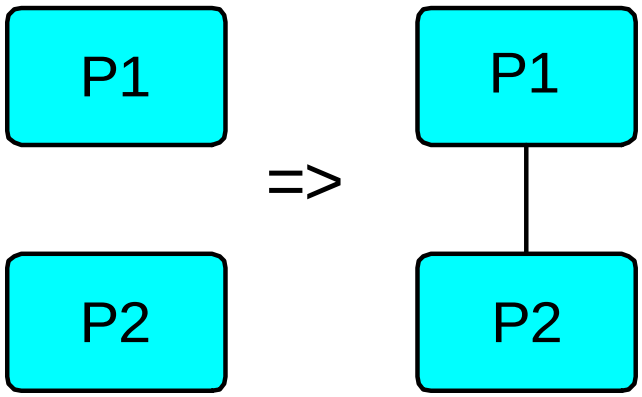


Figure 50: Sequential Operation BSEQ.

MCC-V2(P1 o P2) = MCC-V2(P1) + MCC-V2(P2)

Extensive Structure (Ratio Scale)

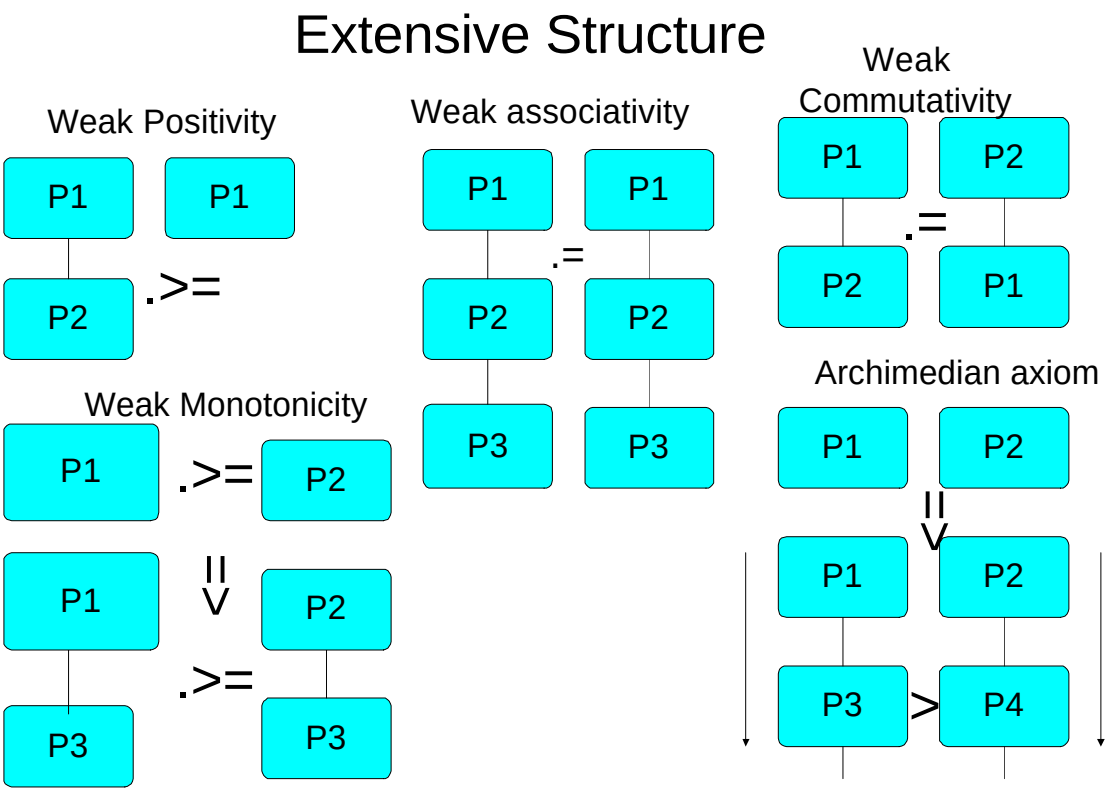


Figure 51: Extensive Structure.

A theorem of measurement theory says that it is sufficient to validate the ranking order in order to validate a measure as a ratio scale.

Definition 2 (Prediction System)

A prediction system consists of a mathematical model together with a set of prediction procedures for determining unknown variables and parameters.

Definition 3
(Validation of a Prediction System)

Validation of a prediction system is the usual empirical process of establishing the accuracy of the prediction system in a given environment by empirical means, i.e., by comparing model performance with known data points in a given environment..

This type of prediction is well accepted in the software measures community.

When people talk about an attempt to validate, for example, the COCOMO-Model or particular software reliability model, they would certainly be attempting the kind of validation described in the definition although they would not know that these are actually prediction systems.

Confusion Between both Types of Validation

There is a confusion between these two separate concepts of validation. What is the reason for it:

It is because of a basic and poorly articulated misconception that a software measure must always be part of a prediction system.

Misconception of Validation

The misconception is normally presented in something like the following manner:

A measure is only valid if it can be shown to be an accurate predictor of some software attribute of general interest, like costs or reliability.

- Suppose that we have some good measure of internal attributes, like size, structuredness, modularity, functionality, coupling, and cohesion.

- Then it is apparently not enough that these measures accurately characterize the stated attributes because these are not considered to be of general interest.

- Since there is generally no specific hypothesis about the predictive capabilities of such measures they are shown to be valid by correlation against any interesting measures which happen to be available as data.

Example

For example, a measure of coupling might be claimed to be valid or invalid on the basis of a comparison with know development costs if the latter is the only data available.

This would be done even though no claims were ever made about a relationship between coupling and development costs.

External Validation

- It is conceivable that a measure could be shown to be valid in the sense of it being a component of a valid prediction system even though no hypothesis existed.

- For this to happen the data happens to be available would have to be shown to be consistently related via a formula determined initially by regression analysis.

- If such validation does occur let us call it **external** validation of the measure to distinguish it from internal validation which should initially take place to show that it actually measures some attribute.

Definition 3 (External Validation)

External validation of a software measure u is the process of establishing a consistent relationship between u and some available empirical data purporting to measure some useful attributes.

Given our poor understanding of the relationship between various software products and processes, external validation seems highly unreliable.

And yet we are expected to accept that this as the major approach to validate.

It may be of some comfort to the many software researchers who have taken the above approach to validation of software measures, to note that they have not been alone in making the same mistakes. And speaking quit generally about measurement validation.

External Validation

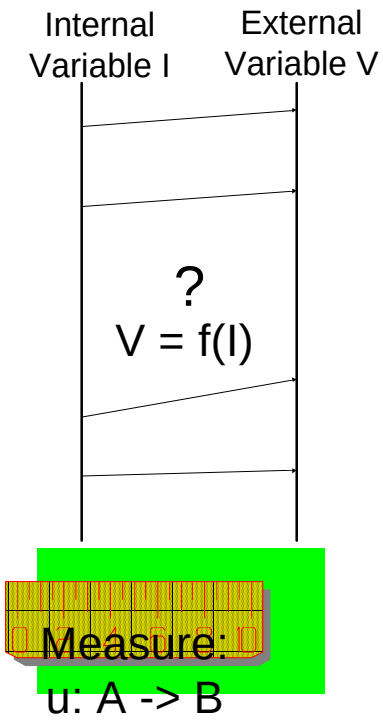


Figure 52: Internal and External Variables.

Internal Variable:	Measure u
External Variable:	Costs of maintenance, etc.
Function f	Function, which has to be validated.

Questions:

1. What is the function f?
2. What properties of an external variable can be predicted?

Function f

Assume it holds:

1.Measure u assumes a ratio scale.

2.External variable V assumes a ratio scale

Function f which has to be validated:

$$V(P) = a\ LOC(P)b,$$

with a,b>0. =

Surprising Result

Having two ratio scales the formula of the basic COCOMO-Model has to be validated.

LOC as a Ratio Scale

LOC as a Ratio Scale can be used as a ratio scale because it assumes an additive combination rule which implies the assumption of an extensive structure.

External Variable V as a Ratio Scale

An external variable assumes can be used as a non-additive or additive ratio scale if it assumes the following combination rule: an Such a combination rule has the following structure:

$$u(P1 \text{ o } P2) = (u(P1)^{1/b} + u(P2)^{1/b})^{b}.$$

It holds:

b = 1 Additive combination rule -> extensive structure

b > 1 Supra-additivity - Wholeness, non-additive combination rule -> extensive structure

b < 1 Sub-additivity, non-additive combination rule -> extensive structure

Calibration

The factor b of the COCOMO-Model is a calibration factor.

Definition (Calibration)

Citation of Krantz et al. /KRAN71/:

A recurrent temptation when we need to measure an attribute of interest is to try to avoid the difficult theoretical and empirical issues posed by fundamental measurement by substituting some easily measured physical quantity that is believed to be strongly correlated with the attribute in question: hours of deprivation in lieu of hunger, skin resistance in lieu of anxiety, milliamperes of current in lieu of aversiveness, etc. Doubtless this is a sensible thing to do when no deep analysis is available, and in all likelihood some such indirect measures will one day serve very effectively when the basic attributes are well understood, but to treat them now as objective definitions of unanalyzed concepts is a form of misplaced operalism.

It must be stressed that we (and we are sure Krantz) are not claiming that measurement and prediction are completely separate issues. On the contrary we fully support the observations of Kyburg:

Citation of Kyburg /KYBU84/.

If you have no viable theory (prediction system) into which X enters, you have very little motivation to generate a measure of X.

Summary of Validation and Prediction

In order to develop guidelines for designing useful software measures we recommend the following:

- We need guidelines for defining software measures. Here the empirical and numerical relational systems of measurement theory are very helpful.

- Guidelines for validating a software measure externally and internally. Here measurement theory is very helpful.

- It is no question that there is of course still much do in the understanding of how to routinely develop reliable and understandable software measures.

- We are sure that measurement theory can help to give theoretical and foundation issues in actually defining and standardizing software measures.

- The real world never seems to work exactly as it should according to theory. In light of this axiom, one of the perhaps unexpected benefits of moving from philosophy to practice is that it forces us to understand the philosophy and theory more clearly.

- When a theory says that a certain phenomenon should (or should not) happen but it does not (it does), then we must re-evaluate our theory.

- Thus, one of the future tasks in this process of moving to practice is the continual testing and the fine tuning of theory.

5 Object-Oriented Environment

6 Results

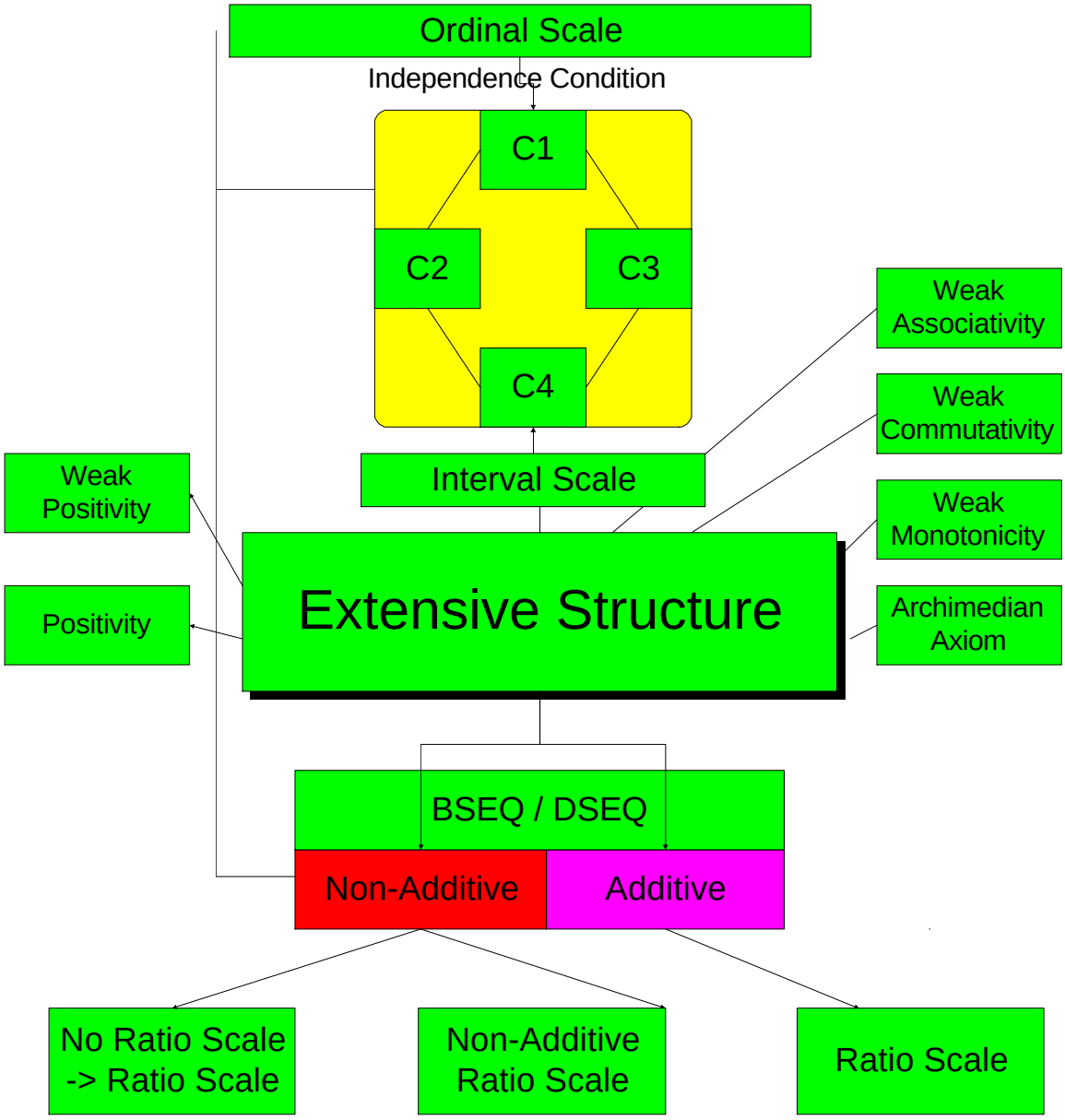


Figure 53: Framework of Software Measurement.

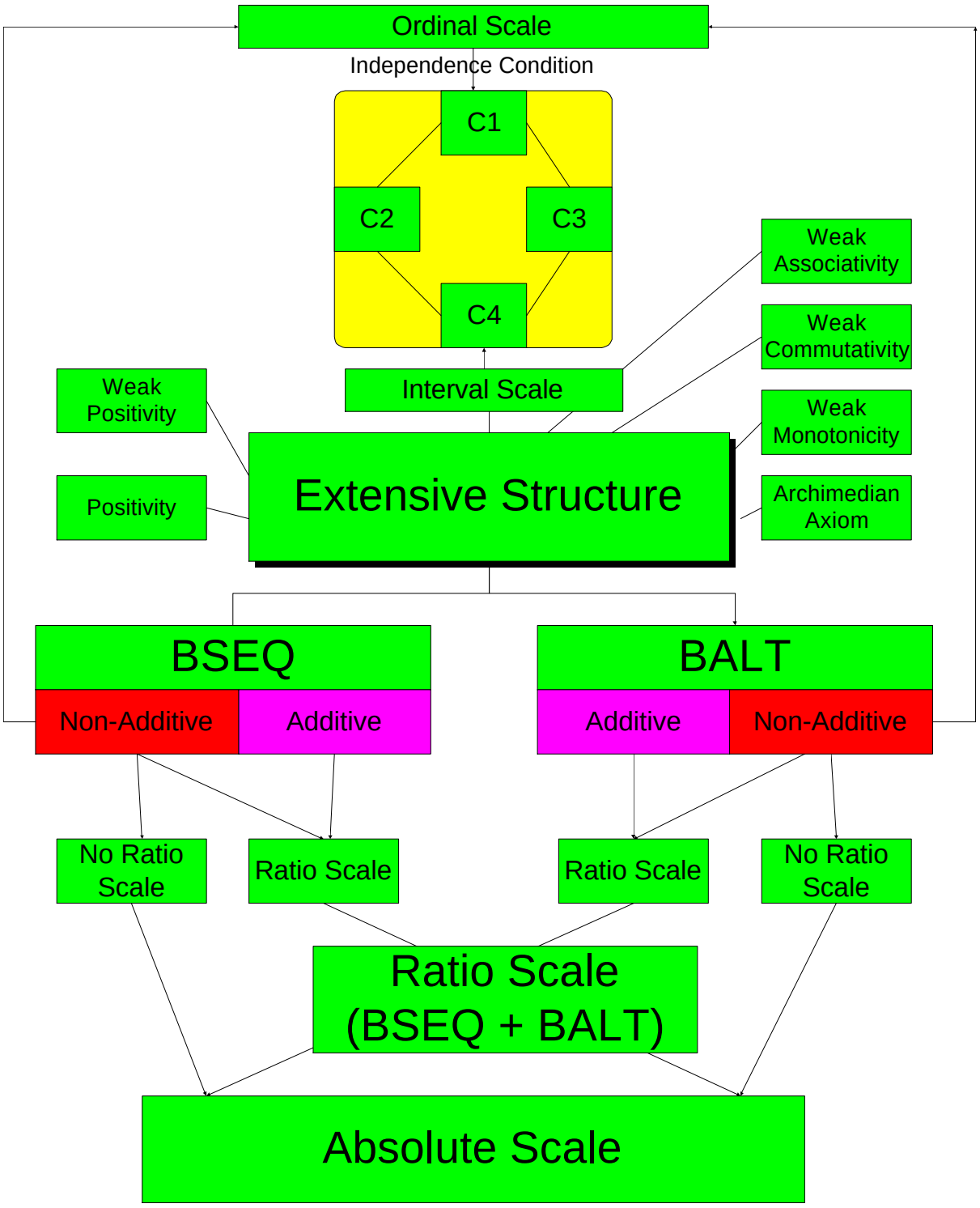


Figure 54: Framework of Software Measurement.

Results (Cont.)

1. Measurement theory gives a clear definition of a measure by a homomorphism between the empirical world and the world of numbers.

2. Measurement theory explains the meaning of numbers by the empirical and numerical relational systems.

3. Measurement theory allows a translation of numerical properties of measures back to empirical properties and vice versa under the assumption of a homomorphism. This helps to understand the properties of measures much easier.

4. Measurement theory shows that the extensive structure plays a central role in software measurement. Measures which assume an extensive structure or not have completely different properties.

5. The independence conditions characterize essential properties of measures related to teamwork and substitution operations.

6. Measurement theory gives conditions for the use of measures on certain scale levels, like nominal, ordinal, interval, ratio and absolute scale. This is important for the proper use of statistics. It also shows, that counting is not an absolute or ratio scale per se.

7. Measurement theory gives conditions for the internal and external validation of software measures. It shows the limits of validation of software measures.

8. Measurement theory gives conditions for prediction models. Here the independence conditions play an important role. They discuss the question whether a software quality attribute of a whole software system can be determined by the components of the system.

9. Measurement theory shows, that we can have additive and non-additive ratio scales.

10. Measurement theory discusses combination rules between concatenated objects. These combination rules give a deeper understanding of the meaning of numbers and give criteria for normalization rules of measurement values.

11. Measurement theory gives clear conditions for verbally formulated (desirable) properties of software measures and shows contradictions. Put in other words: not every combination of conditions is measurable.

12. Measurement theory shows, that the basic COCOMO-Model is the only one which can be used in the case of prediction if we assume ratio scales.

13. Measurement theory derives hypotheses about reality. This is essential, because hypotheses about reality is one of the major goals in science.

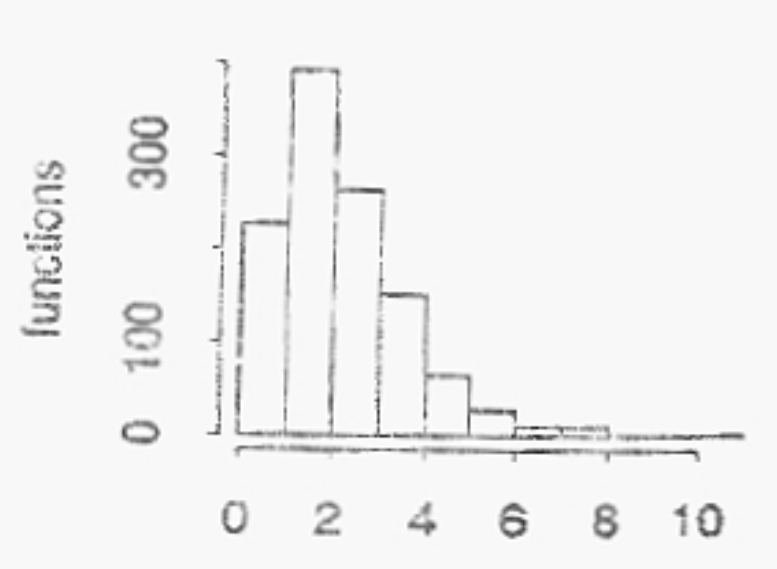
14. Measurement theory explains the idea of size measurement behind the Function-Point Method.

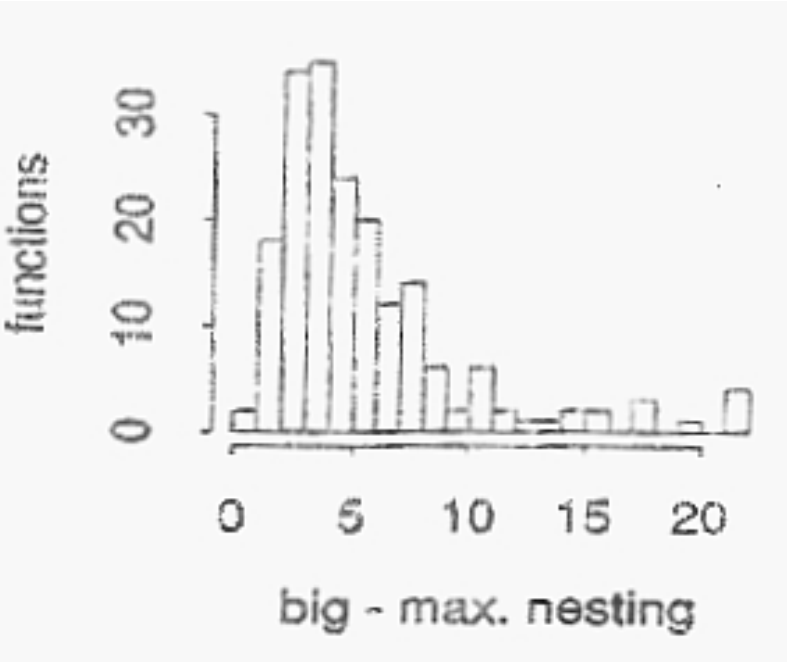
15. Measurement theory gives conditions for hybrid measures.

16. Measurement theory shows that wholeness (The whole is greater than the sum of the parts) is only a numerical modification without changing the empirical evidence of a measure.

17. Measurement theory gives a clear definition of calibration of software measures.

Results (Cont.)





Results (Cont.)