# Properties

# of

# Object-Oriented Software Measures

Horst Zuse

Technische Universität Berlin
Fachbereich Informatik
Franklinstraße 28/29
FR 5-3
10587 Berlin
Gemany
E-mail: zuse at cs.tu-berlin.de
Phone: +49-30-314-73439
Fax: +49-30-314-21103
WWW: www.cs.tu-berlin.de/~zuse

**Keywords**

Software measures, software measurement, measurement theory, object-oriented software measures, concatenation operations, Dempster-Shafer Function of Belief, Kolmogoroff axioms, DeFinetti axioms, Dominance axiom.

**Abstract**

**In this paper foundations of the properties of object-oriented software measures are presented. The criteria for the properties of object-oriented software measures are characterized with several binary operations between objects, classes, methods, etc. Binary operations can be used as a tool to give numbers an interpretation above the ordinal scale level. The result of this investigation is that software measures for object-oriented mostly do not assume an extensive structure. In order to get qualitative criteria for object-oriented measures, the *Dempster-Shafer Function of Belief, the Kolmogoroff axioms* and the *DeFinetti axioms* are introduced. These axioms give qualitative criteria for the use of object-oriented software measures between the weak order and the extensive structure.**

## 1 Introduction

In literature more than two hundred software measures /FETC95/ for applications in the area of object-oriented programming can be found. This process of defining new measures is not finished today. The reason for such a lot of measures may be problems in understanding and maintaining object-oriented programs.

In order to show properties of software measures, we use measurement theory which allows us to translate mathematical / numerical properties of measures back to empirical (intuitive) properties and vice versa. We can do this under the assumption of a homomorphism. In the past we used the extensive structure and the independence conditions /ZUSE92/ to characterize software measures with empirical properties. This concept leads, among others, to the characterization of scale types.
.
In the area of object-oriented software measures the concept of the extensive structure cannot be applied in the same way as for imperative languages. The reason is that the extensive structure is not assumed by many measures in the object-oriented environment. In order to have criteria for properties of measures above the ordinal level for object-oriented software measures, we introduce weaker axioms than the axioms of the extensive structure. We introduce the

Function of Belief, the Kolmogoroff axioms and De Finetti axioms.

The paper is structured as follows: In Section 2 we introduce some aspects of measurement theory, in Section 3 we discuss properties of object-oriented measures with a new axiom system derived from the function of belief, in Section 4 we summarize the results, in Section 5 conclusions are given, in Section 6 follows an attachment related to measurement theory, and Section 7 contains the used references.

## 2 Measurement Theory

There exists a theory, called measurement theory, which gives conditions how to combine empirical conditions with numerical conditions. Measurement theory is also a proper theory to translate mathematical properties of measures back to empirical properties under the assumption of a homomorphism. Readers who are interested to get a deeper understanding of measurement theory we refer to Roberts /ROBE79/, to Krantz et al. /KRAN71/, Luce et al. /LUCE90/ and to some works of the author, like /ZUSE91/, /ZUSE92/, /BOLL93/, /ZUSE94b/. and the Attachment in Chapter 6.

In the past we investigated software measures with measurement theory. We introduced empirical relations •≥ and called them, among others, as *equally or more difficult to maintain.* Mainly, we used the Theorem of the extensive structure to derive empirical conditions from the additive property of a measure. The theorem of the extensive structure in measurement theory says: a Measure u, which is additive related to a binary / concatenation operation, assumes an extensive structure. An extensive structure consists of a set of empirical conditions also called axioms. The extensive structure consists of the following six axioms:

1. Weak order.
2. Axiom of weak positivity.
3. Axiom of weak associativity.
4. Axiom of weak commutativity.
5. Axiom of weak monotonicity.
6. Archimedian axiom.

These six necessary and sufficient empirical conditions are, for example, assumed by the Measure LOC. We illustrate this with a picture.

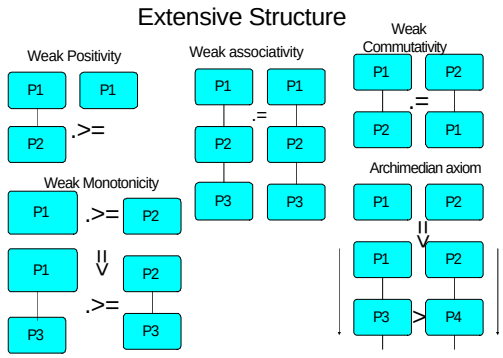


**Figure 2.1:** Extensive structure for flowgraphs and a sequential concatenation operation.

Every Measure u, which is additive, assumes an extensive structure. The approach of the extensive structure also can be applied to cost estimation models, design and maintainability measures, etc.

### 2.1 Extensive Structure and Imperative Languages

In the area of structured programming, imperative languages, software design, etc., we can find many software measures, which are defined as additive measures. Examples are LOC= |N|, and the Measure of McCabe: MCC-V2 = |E| -|N| + 1. From the extensive structure we come easily to the additive ratio scale, and with a modification of the numerical relations system we also come to the non-additive ratio scale.

Put in other words. Using the concept of the extensive structure, the meaning of the numbers of the measures could be interpreted by the empirical conditions of the extensive structure. The extensive structure is the appropriate tool to characterize software measures above the ordinal scale level.

### 2.2 Extensive Structure an Object-Oriented Applications

However, in the area of object-oriented applications, software measures mostly are not additive related to a concatenation operation. Mostly, the Archimedian axiom, and/or the axiom of weak monotonicity are violated. We illustrate this with a picture.

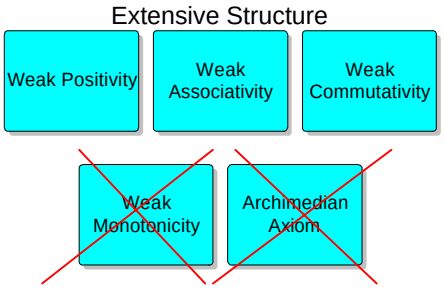


**Figure 2.2:** No Extensive Structure for the most object-oriented measures.

In order to discuss measurement structures in the object-oriented area above the ordinal scale level, we consider qualitative probabilities, quantitative probabilities, and belief structures or belief functions.

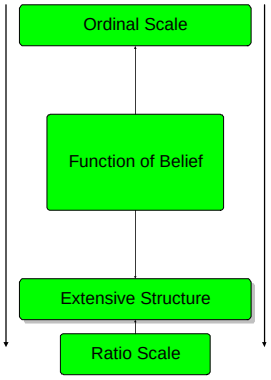


**Figure 2.3:** Ordinal scale, modified function of belief, and extensive structure.

The picture above shows the goal of the new axiom systems. Having only an ordinal scale level, we have a simple level of measurement structures. We are far away from the empirical conditions of the extensive structure. The modified axioms of the function of belief shall help to get more empirical conditions for the interpretation of measurement values in the direction of the extensive structure.

# 3 Measurement Theory and Object-Oriented Software Measures

In this section foundations of the properties of object-oriented software measures are presented.

## 3.1 Introduction

The criteria for the properties of object-oriented software measures are characterized with several concatenation operations between objects, classes and methods. Concatenation operations can be used as a tool to give numbers an interpretation above the ordinal scale level. The result of this investigation is that software measures for object-oriented techniques have completely other properties than measures for imperative languages. It is shown that many of the measures in the object-oriented programming area follow the *Dempster-Shafer Function of Belief* in Artificial Intelligence and the *DeFinetti axioms*.

## 3.2 Abstraction Levels

In order to discuss object-oriented software measures, we have to discuss the abstraction level which we use for measurement in the object-oriented area. For example, the Measure of McCabe is based on the abstraction level of a flowgraph. We discuss for our investigation four levels of abstraction in the area of object-oriented measures.

### 3.2.1 Classes

The first level of abstraction is the class level. A class describes the properties of objects with attributes (often called instance variables) and methods. A class has a set of attributes and a set of methods defined in it. In Figure 3.1 the abstraction of a class as a picture is presented.

#### 3.2.1.1 Methods

The behavior of an object is characterized by the methods defined in its class. A method contains the code of an OO program. Methods can be seen as being subroutines, which are invoked by messages sent to an object. In some OO-systems methods are constructed similar as subroutines in structured programming. The idea, in our case of measurement, is therefore to treat methods as procedures in software measurement.

A method in our view (measurement view) consists of a sequence of statements with a single entry point and a single exit point. The sequence of statements is executed each time the method is called. That means, we can consider the control flow (flowgraph) of the method.

Additionally, a method can be parameterized, having a list of parameters. Parameters can be input, output or both. So we can consider some data flow, additionally. In Figure 3.1 the abstraction of a class with methods is presented.

### 3.3 Structures On Classes

Of course, classes are not the major structure in OO-systems. Classes are related to each other by two types of relationship: *uses relationships* and *inheritance relationships* /BOOC91/. Two other types of relationship will not be considered here, namely *meta classes* and *generic classes*. These constructs are not under consideration by any proposed software measure we found.

#### 3.3.1 Inheritances Relationships

A class A inherits the properties / attributes of class B. The properties of Class B become additional properties of Class A without being defined in Class A again. In turn, Class B is a generalization of Class A.

In Figure 3.1 we have a class vehicle that is able to move. Car and bicycle are specialization's of vehicle, both are able to move too. The method move and the attribute wheels are inherited along the thick arrow from vehicle to its subclasses. These properties are not denoted for the inheriting class.

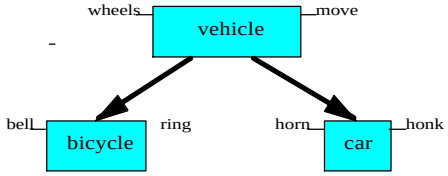


**Figure 3.1:** Classes, methods, and Inheritance.

We see, that we represent a class by a rectangle. The inheritance is represented by a thick arrow, which points to the class which inherits. Methods are represented by a thin line assigned to the rectangle and a name of the method. The specialized classes have additional properties, i.e. a bike can ring it's bell. A class can inherit properties from more than one class. This case is called multiple inheritance and is shown in Figure 3.2

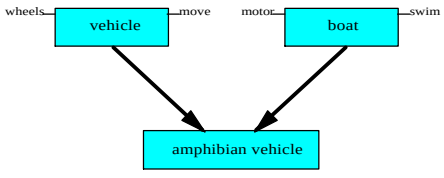


**Figure 3.2:** Multiple inheritance.

Here, the class amphibian vehicle is a special case of vehicle and of boat. So an amphibian vehicle can move and swim.

#### 3.3.2 Uses Relationships

A Class A uses class B, if Class A accesses objects of Class B. Let us consider a Class library with a method borrow a book to borrow a book, where books is a class itself, then the Class book uses the Class Books. The use of a class can be hidden by the implementation, the Class Library could use a Class List, without the appearance in the interface of Class Library. Booch divided these two cases into *uses for interface* and *uses for implementation*. We denote the uses relationship with a thin arrow to the used class and leave the kind of use unspecified.

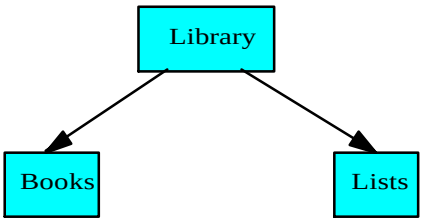


**Figure 3.3:** Uses relationships.

### 3.4 OO-System Level

Taking classes and the two relationships on classes (inheritance and uses relationship) together we can speak of an OO system. That is a set of classes and the uses and inheritance relation on this set.

Typically, none of the two relations builds a complete graph on the set of classes. In contrast, both can build a couple of unconnected subgraphs in one system. In order to be able to cope with structures on classes, we define two subsets of systems. These are

a) *inheritance hierarchies* with one single root. The graph has to be connected, multiple inheritance is allowed.
b) *uses hierarchies* with one single main class that is the only class that is not used by any other class in that graph. The graph must be acyclic and connected.

A system can consist of several of such subsystems. We discuss the properties of measures related to the constructs as described

above. Both inheritance and uses hierarchies contain the single class as a special case.

## 3.5 Concatenation Operations for Object-oriented Programs and their Properties

We now consider for each level of abstraction as described above, concatenation operations. We extend the approach to classes. To be applicable as operations in measurement theory, these concatenation operations have to be complete, which leads to some restrictions on the set of empirical objects on system level. Before discussing the operations themselves, we shall speak of some properties of operations in general.

### 3.5.1 Extensive Structure

Above, we defined the extensive structure which consists of the axioms weak positivity, weak associativity, weak commutativity, weak monotonicity and the Archimedian axiom. For our considerations of object-oriented measures we need a further axioms called idempotency.

**Idempotency**

Idempotency is defined as:

**Definition 3.1: (Idempotency)**
A concatenation rule is called idempotency, if for all objects $a \in A$ holds:

$$a \circ a = a.$$

®
When ever a concatenation operation is idempotent, there is **no way** that the Archimedian axiom can be fulfilled. The consequence is that we have no extensive structure. Hence, having idempotency, we cannot come to the ratio scale via a concatenation operation. Some of the concatenation operations discussed in the following sections are in fact idempotent, so that we have to consider other structures than the extensive structure. These new structures are *belief structures* and the *De Finetti axioms.*

### 3.5.2 Concatenation Operations For Methods

Since many OO programming languages are imperative in their methods, it is possible to apply traditional intra-modular metrics on method level. More than ninety of such measures based on flowgraphs were investigated in /ZUSE91/ /ZUSE92/ together with the sequential and alternative concatenation operations BSEQ and BALT. These concatenation operations were also investigated in this Chapter. A minimal set of software measures for methods is discussed in /ZUSE91/.

### 3.5.3 Concatenation Operations on the Class Level

Chidamber et. al. introduce in /CHID91/ and /CHID94/ a concatenation operation for classes. They use concatenation operations in order to study the properties of their measures. However, the authors base their investigation not on the basis of the extensive structure in order to see whether the measures assume a ratio scale. They consider the Weyuker /WEYU88/ properties. It should be mentioned here, that the Weyuker are not compatible /ZUSE91/, Chapter 6.

We will use the concatenation operations of Chidamber et al. for our studies, too, and we consider some additional concatenation operations.

#### 3.5.3.1 CUNI: Class Unification

The unification of two classes has no means of expression in OO models. As necessary for being an concatenation operation the result of combining two classes is one single new class. This new class combines all the properties the two single classes. If we concatenate Classes A and B to a new class C, we denote this with

$$C=CUNI(A,B).$$

Doing this, we unify the sets of properties of the both Classes A and B to the set of properties of the new Class C. Properties of a class are it's attributes (or instance variables) and methods. The *sets* of properties are *unified*, so identical methods or attributes occur only once. Considering the Classes A and B in Figure 3.4 we see, that both classes have the variable a and method $M_1$ in common. The new Class C has these variable and method only once.

.

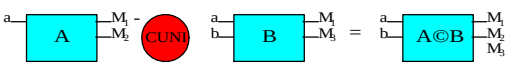


**Figure 3.4**: Class unification.

We now consider the implications of concatenation operations on the inheritance relation.

**Inheritance Relationship**

Within a class declaration we usually find information on (direct) superclasses. We don't find information on subclasses, as we can freely define subclasses to existing classes. We can consider subclasses to some extent to be a form of a property of a class, as for example for polymorphism. Furthermore, some measures proposed in literature use subclasses to characterize a class. For this reason, we consider the inheritance relation in both directions.

We require that the combined class inherits all the properties which any of the two original classes inherited. So any superclass of either A or B or both must be a superclass of the concatenation operation CUNI(A,B). This can lead to multiple inheritance. Consistently, we define all the direct subclasses of A and B to be direct subclasses of the unified class. To illustrate this definition, we consider some cases including those discussed in /CHID94/
.
The first case discussed by Chidamber et. al. assumes two Classes A and B having the same direct superclass (father) V. The resulting class has the same father V. The direct subclasses (children) of A and B respectively also are children of the unified class. Figure 3.5 illustrates this case.

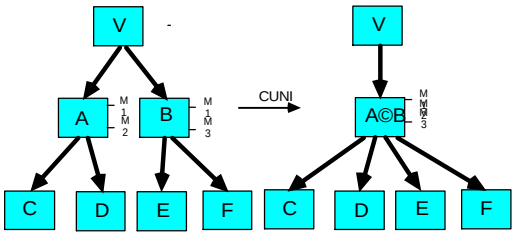


**Figure 3.5**: Unification CUNI (A,B) of sibling classes.

If Class B is a direct subclass of class A, these will be combined as follows. The father V of A also was an indirect superclass of B and is now the father of CUNI(A,B). The children are again all children of A and B, in this case the children of B do not inherit more properties as they inherited already the properties of A via B (See Figure 3.6).

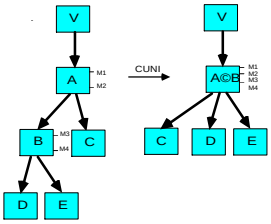


**Figure 3.6:** Unification CUNI(A,B) with a direct subclass

Classes that have separate fathers have to be combined as shown in figure Figure 3.7. The combined class inherits from both superclasses via multiple inheritance. The children are again all the children of A and B, here C multiply inherits from A and B, of course it is only once a child of CUNI(A,B).

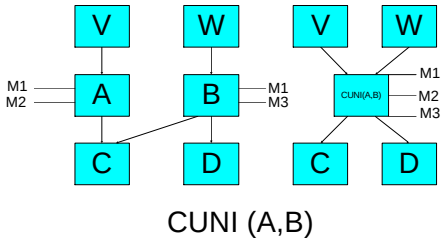


**Figure 3.7:** Unification CUNI(A,B) of unrelated classes.

We now consider the implications of CUNI for uses relations. The behavior is very similar as described above for inheritance. All the classes used by either one ore both of the classes combined will be used by the unified class. This lies already in the unification of the properties defined in the class. Analogously, we define all classes that used any single or both separate classes to use the combined class. This won't cause problems as those classes need not use the additional parts of the new class but the original parts of that class already used. The figures below illustrate this case with two examples.

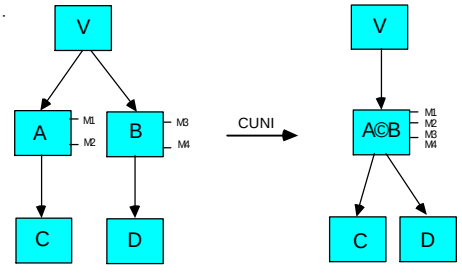


**Figure 3.8:** Concatenation operation CUNI(A,B) with uses relationships.

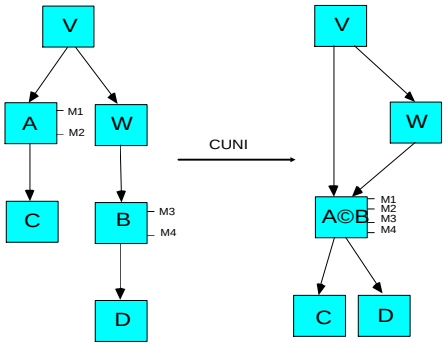


**Figure 3.9:** CUNI with uses relationships

We now consider the properties of the concatenation operation CUNI.

**Properties of CUNI**

Since the unification of the sets of attributes and the sets of methods are independent of the ordering in which these sets are unified, and no order is defined neither for the uses nor inheritance relation, CUNI is commutative and it is also associative. Interesting is, that CUNI is also idempotent: if A is a class it holds:

$$A \circ A = A$$

Assuming idempotency by a concatenation operation has the consequence, that the a measure does not assume an extensive structure which has the consequence that we cannot come to the ratio scale via the extensive structure.

Of course, this property of CUNI is not by an accident. CUNI has been defined as a *unification* and we know that unification of sets is idempotent. This reflects an advantage of object-oriented design. Similar classes should be combined to one class with the consequence of less maintenance effort.

Chidamber et. al. check, whether their measures agree with the Weyuker properties. If we keep in mind, that CUNI is idempotent, it is not surprising that none of their six measures agree with wholeness, which is defined as:

$$u(A \circ B) \geq u(A) + u(B),$$

for all $A, B \in$ **A.** If idempotency is fulfilled, we cannot have wholeness, as Weyuker requires that. Wholeness requires an non-additive ratio scale. In fact, if A=B, we obtain for any Measure μ

$$u(CUNI(A,A) = u(A).$$

As we mentioned above, this is due to CUNI is a kind of an unification. Our suggestion is to add a second combination operation that corresponds to intersection of sets.

### 3.5.3.2 CINT: Class Intersection

We have seen, that the unification of classes leads to a class which includes all the properties, methods and attributes which are common for the both single classes.

The result of an intersection of two classes is that the common properties of both classes are reduced to one property in the combined class. We illustrate this with the following picture. Assume two Classes A and B with variables and methods. The new Class C, combined by Classes A and B only consists of the variable a and method M1.

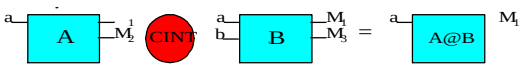


**Figure 3.10**: CINT: Intersection of classes.

As for CUNI, we now consider the implications of CINT for the inheritance relationship. It is important to say that CINT is not a combination rule to build new stand alone classes. We are discussing the consequences for a concatenation operation which is defined as CINT. Later we will discuss generalization, then we will see an application of CINT in reality.

CINT intersects the sets of properties of classes. We now require that the intersected class inherits all the properties that both classes inherited. These properties are again common to both classes. And we require that every subclass that are common to both classes will be subclasses of the intersection.

That leads to somewhat non-intuitive situations as shown in Figure 3.11. Here, G is an indirect superclass of A and B and so the intersected class CINT (A,B) will inherit from G but neither from V nor W. And the only common subclass is E, which becomes a child of the intersection.

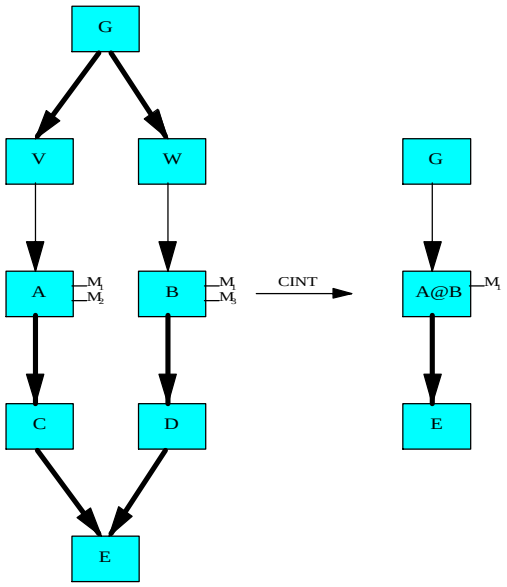


**Figure 3.11:** Class intersection and inheritance relationships.

Again, for the uses relation our consideration are very similar. Only those classes which use both classes use the intersection and as an implication of the intersection of properties. Only those classes used by both classes will be used by CINT(A,B). Unlike inheritance indirect use does not propagate along the vertices of the uses relation.

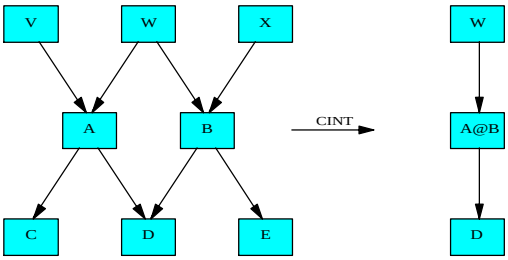


**Figure 3.12:** Class intersection and uses relationships.

We now have two combination operations CUNI and CINT among classes and we find a couple of measures including some of the measures proposed by Chidamber et. al. holding the following condition

$$\text{C-NOM(CUNI (A,B))} = \text{C-NOM(A)} + \text{C-NOM(B)} - \text{C-NOM(CINT(A,B))},$$

where C-NOM is the number of methods in a class. This equation is well known in probability theory, describing a probabilistic measure (Third Kolmogoroff axiom, see below).

### 3.5.4 The Empty Class

We now define the empty Class ∅ to be a class with no properties, i. e. empty sets of attributes and methods. Furthermore, the empty class may not inherit any property and no class inherits from ∅ . It is also not used by any class and of course does not use any class.

Having defined an empty class we can say that two classes A and B are disjunct, i. e. they don't have any property in common. It holds

$$\text{CINT}(A,B) = \varnothing$$

Another axiom for measures is the following:

$$u(\varnothing)=0,$$

which again corresponds to probability. The correspondence shall not lead to the assumption that software measures would in any way be measures of uncertainty in OO. In fact, one of the major properties of probability functions, the limitation to the Interval [0,1] is not adhered by the measures in general and is not our intention, too.

But the observation that measures in combination with the two operations described here (partly) are similar to probability measures, is useful.

With this support, we can derive additional properties of the preference relations induced by the measures. These conclusions can be drawn, because probability theory has examined the relations between qualitative and quantitative probability. If these axiomatic systems are considered carefully under the circumstances of OO measurement, it comes up that these axioms can be adapted to the unlimited case.

## 3.6 Belief Structures

As we have shown above, measures which are additive, assume an extensive structure. They also can be used as an additive ratio scale. Measures which are not additive, can assume an extensive structure, too. Such measures are a strong monotonic function of the additive measures. Non-additive measures, which assume an extensive structure, can be used - under certain conditions- as a non-additive ratio scale. It should be mentioned here that measures can assume an extensive structure

but cannot be used as a ratio scale. This is the case if the combination rule is not meaningful for the admissible transformation of the ratio scale. The Measure MCC-V = |E| - |N| + 2 is such a case.

In the object-oriented environment the behavior of measures related to concatenation operations is different. Many measures do not assume an extensive structure which has the consequence that they cannot be used as an (additive) ratio scale. Other measures assume idempotency, which implies, that they do not assume the Archimedian axiom, which implies no extensive structure, either.

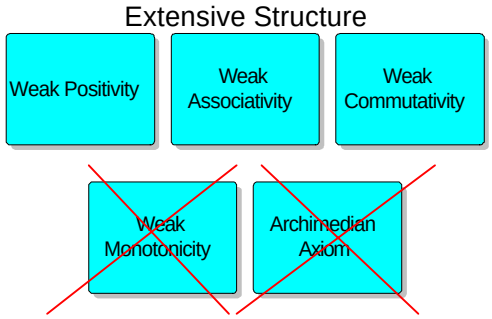


**Figure 3.13:** No Extensive Structure for the most object-oriented measures.

In order to discuss measurement structures in the object-oriented area above the ordinal scale level, we consider qualitative and quantitative probabilities and belief structures or belief functions.

### 3.6.1 Belief Functions

There exist a well known quantitative approach, the theory of belief functions, which has generated considerable interest in recent years /SHAF76/, /SHAF87/. In this theory, belief may be interpreted as a generalization of probability. Also belief functions provide a useful and effective tool for the quantification of subjective, personal judgments. It is the view of many researchers that humans frequently reason in qualitative rather quantitative terms /BHAT86/. A function of belief describes the belief in a hypothesis and the belief in combination of hypotheses. More information's about belief functions can be found in /SHAF76/, /BUER87/ and /WONG91

Firstly, we consider the Kolmogoroff axioms and then as a generalization, the function of belief.

#### 3.6.1.1 Kolmogoroff Axioms

It is important to mention that we treat the Kolmogoroff axioms as (part) of a representation theorem. Put in another way, we shall attempt to treat the assignment of probabilities to events as a measurement problem of the same fundamental character as the measurement of, e.g. mass or length.

Typically, probabilities are considered in an interval [0,1]. The approach of Kolmogoroff describes events as subsets in a probability space /KRAN71/, p.200.

**Definition 3.2: (Kolmogoroff Axioms (1933))**
*Suppose that X is a non-empty set, that ℑ is an algebra of sets on , and that P is a function from ℑ into the real numbers. The triple <X, ℑ, P> is a (finitely additive) probability space iff, for every A, B ∈ ℑ:*

$P(A) \geq 0$ (K2)
$P(X) = 1$ (K2)
If $A \cap B = \emptyset$, => $P(A \cup B) = P(A) + P(B)$ (K3)
®

Krantz et al. writes: *Our plan, instead, is to treat Definition 3.2 as (part of) a representation theorem; specifically, we inquire into conditions under which an ordering •≥ of ℑ has an order-preserving function P that satisfies Definition 3.2. Obviously, the ordering is to be interpreted empirically as meaning „qualitatively at least as probable as". Put another way, we shall attempt to treat the assignment of probabilities to events as a measurement problem of the same fundamental character as the measurement, e.g., mass or momentum. From this point of view, the debates about the meaning of probability are, in reality, about acceptable empirical methods to determine •≥. It is not evident why the measurement of probability should have been the focus of more philosophic controversy than the measurement of mass, or length, or of any other scientific significant attribute; but it has been. We are not suggesting that the controversies over probability have been justified, but merely that other controversial issues in the theory of measurement may have been neglected to a degree.*

On page 201, we find the following statements: *Ellis point out that the development of a probability ordering is .. analogous to that of finding a thermometric property, which .. was the first step towards devising a temperature scale.*
*The comparison between probability and temperature may be illuminating in other*

*ways. The first thermometers were useful mainly for comparing atmospheric temperatures. The air thermometers of the seventeenth century, for example, were not acceptable for comparing or measuring the temperatures of small solid objects. Consequently, the early history of thermometry, there were many things which possessed temperature which could not be fitted into an objective temperature order. Similarly, then, we should not necessarily expect to find any single objective procedure capable of ordering all propositions in respect of probability, even if we assume that all propositions possesses probability. Rather, we should expect here to be certain kinds of propositions that are much easier to fit into an objective probability order than others.*

We will see that certain measures have a similar behavior related to concatenation operations as probabilities. It is important to say that objects of software are not probability events. We consider the definition above in the context of mathematically axioms and as a representation theorem. We consider the assignment of probabilities to events as a measurement problem. The relationships do not base on probability events.

In the following, we give a generalization of quantitative functions of probabilities, and that is the function of belief and their relationship to a qualitative belief. Then, these structure of belief will be modified, that we can apply this axiom system to object-oriented software measures.

### 3.6.2 Belief Functions of Dempster and Shafer

The functions of belief by Dempster and Shafer /SHAF76/ are a tool for the description of uncertain knowledge. An detailed discussion of functions of belief can be found in /BUER87/. The axiomatic of a function of belief is manly different in the third axiom of the Kolmogoroff axioms (KB3).

We now consider the function of belief.

**Definition 3.3**: **(Function of Belief):**
*Suppose X is a finite set, then a belief function Bel is a mapping from: $2^X$ to the interval [0,1]. Then a function Bel: $2^X$ -> [0,1]. is a function of belief iff the following axioms hold:*

$$Bel(\varnothing) = 0 \qquad (B1)$$

$$Bel(X) = 1 \qquad (B2)$$

$$Bel(A1 \cup A2 \cup .. \cup An) \geq \sum_{I \neq 0, \langle 1..n \rangle} (-1)^{|I|+1} Bel \bigcap_{i \varepsilon I} Ai \qquad (B3)$$

For two elements A, B $\in 2^X$ means B3:

$$Bel(A \cup B) \geq Bel(A) + Bel(B) - Bel(A \cap B). \qquad (B4)$$

®

Instead of the sign = with the Kolmogoroff axioms we find here a greater or equal. The belief in the unification of two disjunct events $A \cap B = 0$ can be greater than the sum of belief of the single results. Important for us is the part: - *Bel (A ∩ B),* which shows us exactly the behavior of object-oriented software measures.

The question is here what are the consequences if we want to use such an object-oriented measure to predict costs of maintenance. The effort for maintenance would be the sum of the maintenance of both components minus the effort of maintenance which is common in both components. We should mention here that the Function-Point Method shows the same behavior. The reason is that combining two projects to one project, we use, for example, input files only once.

### 3.6.3 Qualitative Belief

As we saw with the discussion of relational systems in measurement theory, we considered an empirical and a numerical relational system. Both were connected by a homomorphism. The empirical relational system describes **qualitative conditions**. **Similar, we can formulate a qualitative belief.** We believe to an event $A \in 2^X$ not less than to an event $B \in 2^X$, if

$$A \bullet\geq B.$$

Let us consider the relation $\bullet\geq$ more in detail. Wong et al. /WONG91/, (See also: /BUER87/, p.38ff) showed the following theorem: There $\bullet>$ is defined as:

$$A \bullet> B <=> A \bullet\geq B \wedge \neg (B \bullet\geq A).$$

**Theorem 3-1:**
*Let X be a set, $\bullet\geq$ a preference relation on $2^x$. It exists a function of belief, which corresponds completely with $\bullet\geq$, that means*

$$A \bullet\geq B <=> Bel(a) \geq Bel(B),$$

*if the following axioms are satisfied*

$\forall A, B \in 2^X : A \bullet\geq B \vee B \bullet\geq A$, completeness (QB1)

$\forall A, B, C \in 2^X : A \bullet\geq B \wedge B \bullet\geq C \Rightarrow A \bullet\geq C$, transitivity (QB2)

$\forall A, B \in 2^X: A \supseteq B \Rightarrow A \bullet\geq B$, dominance (QB3)

$\forall (A \supset B, A \cap C = 0): (A \bullet> B \Rightarrow A \cup C \bullet> B \cup C)$ partial monotonicity (QB4)

$X \bullet> 0$ prevent triviality (QB5)
®

It holds:

- Axiom QB1 is completeness, and
- Axiom QB2 is transitivity. Both required by a weak order.
- Axiom QB3 is the dominance axiom,
- Axiom QB4 is the partial monotonic axiom, and
- Axiom QB5 avoids triviality.

We illustrate this with some pictures.

### 3.6.3.1 Set of all Attributes and Methods

In order to illustrate the new axiom system, we introduce the sets of methods and attributes.

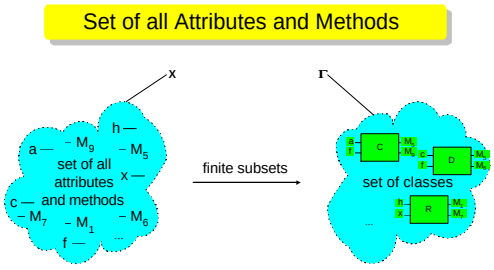


**Figure 3.14:** Explanation of the sets X and the set of classes.

The picture above shows the used sets for our investigation. The left *cloud* shows the set of all attributes and methods. The right *cloud* shows the set of classes, which structure the attributes and methods. The next picture shows the axiom of partial monotonicity (QB4).

### 3.6.3.2 Axiom of Partial Monotonicty of Qualitative Belief

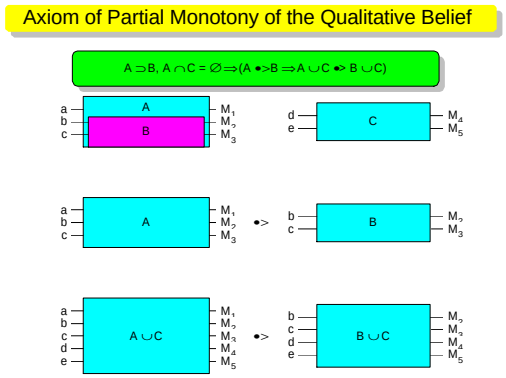


**Figure 3.15:** Partial monotonictiy QB4.

Assume we have a Class A consisting of attributes and methods. Class consists of the attributes a, b, c and the methods M1, M2, and M3. Class B consists of the attributes b, c and the methods M1 and M2. It follows that Class B is a real subset of A. We write this as $A \supset B$.

Furthermore, we have a third Class C, consisting of the attributes d, e, and the methods M4 and M5. We see, that Class C does not have any attributes or methods common with Class A. The intersection of the both Classes A and C is the empty set. We write this as $A \cap C = \emptyset$.

We now assume that we have an empirical relation $\bullet\geq$. We denote this relation as *equally or more difficult to maintain*. Assuming $A \bullet\geq B$, what means Class A is *equally or more difficult to maintain* than B, it follows that $A \cup C$ is also equally or more difficult to maintain than $B \cup C$. We write this as $A \bullet\geq B \Rightarrow A \cup C \bullet\geq B \cup C$. The Classes in Rows 2 and 3 show this behavior. If a user has this view of maintenance of classes then he has to look for a Measure u which fulfills his view. We can write this as:

$$A \bullet\geq B \Rightarrow A \cup C \bullet\geq B \cup C \Leftrightarrow$$
$$u(A) \bullet\geq u(B) \Rightarrow u(A \cup C) \bullet\geq u(B \cup C).$$

We call this a homomorphism related to the axiom of partial monotonicty. It is important to mention that the condition above only holds in the direction of =>. The next picture illustrates the dominance axiom QB3.

### 3.6.3.3 Dominance Axiom

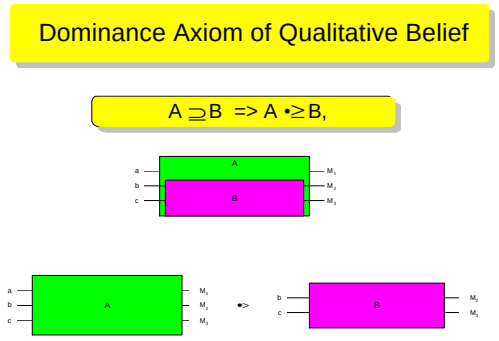


**Figure 3.16:** Dominance axiom.

The dominance axiom shows the following: Assume Class B is a subset of Class A. Then it follows that Class A is equally or more difficult to maintain than Class B. We write this as: A ⊇ B => A •≥ B. This is another important criterion for an object-oriented software measure. We now define the DeFinetti axioms.

### 3.6.4 DeFinetti Axioms

The DeFinetti axioms /ROBE79/, /BUER87/, /DEFI37/, /FETC95/, give necessary, but not sufficient conditions for the existence of a relation of belief. If these axioms are fulfilled, then a function of belief exists, which corresponds with the relation •≥ . It holds:

$\forall A, B \in 2^X : A \bullet\geq B \vee B \bullet\geq A,$
completeness (F1)

$\forall A, B, C \in 2^X : A \bullet\geq B \wedge B \bullet\geq C \Rightarrow A \bullet\geq C,$
transitivity (F2)

$\forall A \in 2^X : A \bullet\geq 0$ (F3)

$\forall ( A \cap C = B \cap C = 0)$: $(A \bullet> B \Leftrightarrow A \cup C \bullet> B \cup C)$, monotonicity (F4)

$X \bullet> 0$ prevents triviality (F5)
®

Here, we see that the De Finetti axiom of monotonicity (F4) is stronger than partial monotonicity (QB4) of the qualitative belief. We illustrate this with a picture.

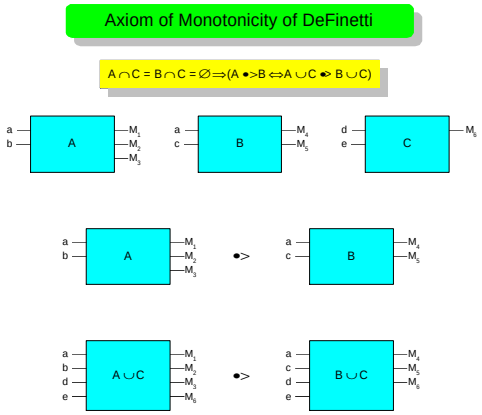


**Figure 3.17:** Axiom of Monotonicity of the DeFinetti axioms.

The axiom of partial monotonicity of DeFinetti is stronger than the axiom of partial monotonicity as discussed above. Assume, we have three Classes A, B and C with attributes and methods. We also assume it holds that the Classes A and C, and B and C do not have common attributes and methods. We write this as: $A \cap C = \emptyset$, and $B \cap C = \emptyset$. Both together we write as: $A \cap C = B \cap C = \emptyset$.

We assume that Class A is equally or more difficult to maintain than B ($A \bullet\geq B$). If holds $A \bullet\geq B$ then we expect that also holds: $A \cup C \bullet\geq B \cup C$. We write the complete axiom as:

$$A \bullet\geq B \Leftrightarrow A \cup C \bullet\geq B \cup C.$$

This axiom holds for both directions. It is illustrated in the picture above in Rows 2 and 3. We now apply the axiom systems above to the object-oriented area.

## 3.7 Belief Function and De Finetti Axioms in the Object-Oriented Environment

It should be made clear again, that software measures in the object-oriented area are not probability measures. We do not interpret the measurement values as probabilities and we do not limit the measurement values to [0, 1]. Nevertheless, we believe, that it makes sense to describe the properties of some software measures with the structures, as presented above. Many measures in the object-oriented area fulfill, with the exception of Bel(X)=1, the axioms of belief structures.

We now modify the axioms that we can apply them to object-oriented software measures. Doing this, we get, without using the extensive structure, statements of empirical relations, which are more than the properties

of the ordinal scale. The axioms of the modified function of belief u are the following:

**Definition 3.4: (Modified Function of Belief)**
*Let X be a countable set and ℑ the set of finite subsets of X. A Measure u: ℑ -> ℜ is a modified function of belief iff*

$u(0) = 0$ (MFB1)

$\forall A \in \Im: u(A) \geq 0$ (MFB2)

$$u(A1 \cup A2 \cup .. \cup An) \geq \sum_{I \neq 0, \langle 1..n \rangle} (-1)^{|I|+1} u\left(\bigcap_{i \in I} Ai\right)$$ (MFB3)

®

In the object-oriented environment the set X represents the set of properties of classes. A class is then a finite subset of these properties. That means, the classes are elements of ℑ. We now modify the axioms for qualitative belief, too.

**Definition 3.5: (Modified Relation of Belief):**
*Let •≥ a relation on ℑ, then •≥ is a modified relation of belief iff,*

$\forall A, B \in \Im: A \bullet\geq B$ or $B \bullet\geq A$ (MRB1)

$\forall A, B, C \in \Im: A \bullet\geq B$ and $B \bullet\geq C \Rightarrow A \bullet\geq C$, (MRB2)

$\forall A \supseteq B \Rightarrow A \bullet\geq B$, dominance axiom (MRB3).

$\forall (A \supset B, A \mid\cap C = 0) \Rightarrow (A \bullet> B \Rightarrow A \cup C \bullet> B \cup C)$, partial monotonicity (MRB4),

$\forall A \in \Im: A \bullet\geq 0$ , Positivity (MRB5).

®

We can take over Theorem 3.1 for the modified structures of belief.

**Theorem 3-2:**
*It exist a modified function of believe, which fulfills (1), (2), (3) of the modified function of belief (MFB1, MFB2, MFB3), such that*

$$A \bullet\geq B <=> u(A) \geq u(B),$$

*if •≥ fulfills the axioms of the modified relation of belief (MRB1, MRB2, MRB3, MRB4, MRB5).*
®

We take over the proof from Wong et al. /WONG91/. The modification has the consequence that we have a no-upper boundary of u. The normalization to the Interval [0,1] is the last step in the proof. Since X is countable infinite, it does not exist a measurement value for X. We only consider finite subsets of X, this holds for u and for •≥. For this reason the complement Ac = X \ A of an object is no element of Γ. For this reason we can not consider here a plausibility function, which are additions for the function of belief. The proof in /WONG91 does not use the complement set.

**Proof of the Theorem**
In order to do not overload the paper, we refer to the proof of the theorem to Fetcke /FETC95/.

### 3.8 Investigations of some Object-Oriented Measures

We now apply the discussed concepts above to some measures in the object-oriented environment. We describe them with the axioms of the modified function of belief. We consider whether a measure fulfills the axioms MFB1-MFB3. In the most cases the measures fulfill the stronger condition

$$u(A \cup B) = u(A) + u(B) - u(A \cap B),$$

which is a sufficient condition for MFB3. We illustrate the statement above with the following picture.

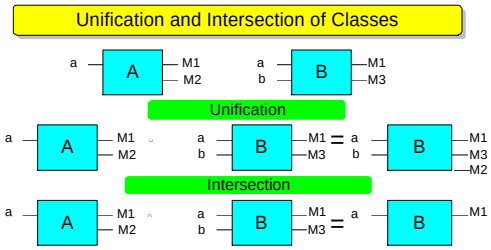


**Figure 3.18:** Unification and Intersection of Classes A and B.

The unification of Class A ∪ B and the intersection of Classes A ∩ B are shown in the picture above. The discussed conditions above are implied by this condition. The condition above says that a measure which follows the condition above is only additive if no common attributes and methods are existing. That is mostly not the case in the object-oriented environment.

On the class level we investigate the measures in the table below. We investigate whether the measures fulfill the axioms of the modified

function of belief. If this is not the case, the axioms of the modified relation of belief are investigated:

MRB1: Is always fulfilled by the properties of the real numbers

MRB2: Is always fulfilled by the properties of the real numbers

MRB3 is the Dominance Axiom,

MRB4 is the partial monotonicity axiom,

MRB5 is the Positivity axiom.

Following Theorem 3.2 a modified function of belief fulfills the axioms MRB1-MRB5.

**Measure CBO**

We consider the following example of the Measure CBO. IT is defined as:

C-CBO is the number of classes that are coupled with the Class C. Two classes are coupled to each other if methods of one class methods or instance variables of other classes use. We illustrate this with an example.

In the picture below Class A is coupled with the four Classes V, W, C, and D. It holds C-CBO(A) = 4.

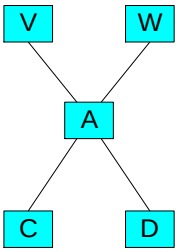


**Figure 3.19:** Class A is coupled with four other classes. Class A is used by the Classes V and W and Class A uses itself the Classes C and D. It holds CBO(A) = 4.

The behavior of this measure can be described with the class intersection. The modified axioms of the function of belief are fulfilled, it holds for the Classes A and B.

CBO (Ø) = 0

For all A: CBO (A) ≥ 0

CBO (CUNI(A, B)) = CBO (A) + CBO (B) - CBO (CINT(A , B))

For the third axiom we need a further example.

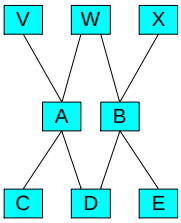


**Figure 3.20:** CBO related to the concatenation operations CUNI and CINT.

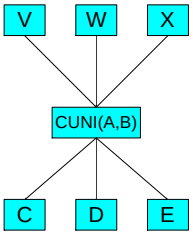


**Figure 3.21:** CBO related to the concatenation operations CUNI and CINT.

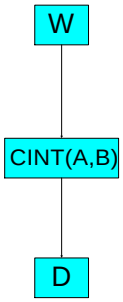


**Figure 5.5:** CBO related to the concatenation operations CUNI and CINT.

It holds:

CBO (A) = 4
CBO (B) = 4
CBO (CINT(A,B)) = 2
CBO (CUNI(A,B)) = 4 + 4 - 2 = 6.

In /FETC95/ more examples are presented. We now show an overview of the properties of the Measures of Chidamber et al.

## 4 Results

We now present the results of our investigation. The Measure CBO is Coupling between Objects, LOC is the Measure lines-of-code for methods, NOC is the Number Of Children, WMC is the Weighted Method Count, Response For a Class, and LCOMa and

LCOMb are the Measures Lack of Cohesion of Methods /CHID91/, CHID92/.

| Measure | G | MRB3 | MRB4 | MRB5 |
|---|---|---|---|---|
| CBO | Y | Y | Y | Y |
| LOC | Y | Y | Y | Y |
| NOC | Y | Y | Y | Y |
| WMC | Y | Y | Y | Y |
| RFC | Y | Y | Y | Y |
| LCOMa | N | Y | N | Y |
| LCOMb | N | N | N | Y |

**Figure 4.22:** The table shows the properties of some object-oriented measures related to the properties of the modified Function of Belief (G) and the modified relations of belief (MRB3), (MRB4), and (MRB5).

The table of object-oriented measures above shows very clearly, that most of the measures do **not** assume an extensive structure. An exception is the Measure LOC. However, considering the modified function of belief and the relations of belief, we can characterize the measures above the level of the ordinal scale. To characterize measures above the ordinal scale level is very important because we can do not very much with ordinal numbers. It means especially here, that we have more sophisticated properties of object-oriented measures.

Other results of our investigation are the following:

1. In order to get qualitative criteria for the use of object-oriented measures, we introduced qualitative conditions derived from the function of belief.

2. In the area of object-oriented measures we can define many concatenation operations, like CUNI and CINT. Concatenation operations are useful to get more information of the meaning of the numbers of measures above the ordinal scale level.

3. We demonstrated that object-oriented measures mostly do not assume an extensive structure which blocks the way to the ratio scale via the extensive structure. The reason is idempotency, which implies that the Archimedian axiom is not fulfilled.

4. We showed the consequences for object-oriented techniques if the investigated measures assume intersection and unification properties.

5. Considering the modified function of belief and the qualitative relation of belief, we can characterize the measures above the level of the ordinal scale. The partial monotonicity axiom, the dominance axiom, and the axioms of DeFinetti are appropriate to give a **qualitative** characterization of object-oriented software measures.

## 5 Conclusion

Mostly, object-oriented measures do not assume an extensive structure. We showed, that in the area of object-oriented measures we can find properties of idempotency in the context of the binary operations unification and intersection of sets. The function of belief, qualitative belief, and the De Finetti axioms are appropriate to give a qualitative characterization of object-oriented software measures above the ordinal scale level. This paper should be seen as a first step to analyze object-oriented measures by new measurement structures.

## 6 Attachment: Foundations of Software Measurement

In this attachment we introduce basic concepts of measurement theory and show the application to software measures. We use the text of Krantz et al. /KRAN71/, Luce et al. /LUCE90/ and Roberts /ROBE79/. Proofs of the theorems and a more detailed discussion of measurement theory related to software measures can be found in /ZUSE91/, /ZUSE92/, /ZUSE92b/, /ZUSE94c/, and /BOLL93/. An application of measurement theory to more than ninety intra-modular software complexity measures can also be found in /ZUSE91/.

It should be mentioned here that it is a widely spread misunderstanding that measurement theory is only useful to determine the scales like nominal, ordinal, interval and ratio scale. This is more a side-effect of measurement theory. The major advantage of measurement theory lies in hypotheses about reality.

### 6.1 Basic Concepts of Measurement Theory

First of all we want to introduce the notation of an empirical, a numerical relational system and a scale. Let

$\mathbf{A} = (A, \bullet\geq, o)$

be an empirical relational system, where A is a non-empty set of empirical objects, •> is an empirical relations on A and o a binary operation on A (Of course, there are more than one relation and binary operation possible).

According to Luce et al. /LUCE90/, p.270, we assume for an empirical relational system **A** *that there is a well-established empirical interpretation for the elements of **A** and for each relation Si of **A***. We also assume the same for the binary operations.

Let further

**B** = (ℜ, ≥, +)

be a formal relational system, where ℜ, are the real numbers, ≥, a relation on B, and + a closed binary operation on ℜ. (Of course, there are more than one relation and binary operations possible). We also include the case that there are no relations or no operations.

The sign + means here the following: for a concatenation of two Flowgraphs P1 o P2 holds the following formula:

$$u(P1 \text{ o } P2) = u(P1) + u(P2) ,$$

where u is a software measure and P1, P2ε **P**

A **measure** is a mapping u: A -> B such that the following holds for all P1, P2 ε A:

$$P1 \bullet\geq P2 <=> u(P1) \geq u(P2)$$
and
$$u(a \text{ o } b) = u(a) + u(b)$$

Then the Triple (**A**, **B**, u) is called a **scale**. According to this definition we see that measurement assumes a homomorphism.

Given two relational system **A** and **B** we can ask whether there exists a measure such that (**A**, **B**, u) is a scale. This problem is called the **representation problem**. If such a measure exists we can ask how uniquely the measure is defined. This problem is called the **uniqueness problem**. The uniqueness problem leads to the definition of scale types such as ordinal or ratio scale.

Let g: X ->Y and h: Y -> Z be mappings. Then hg denotes the composed mapping hg(x)=h(g(x)) for x ε X.

**Definition 6.1: (Admissible Transformation):**

Let (**A**, **B**, u ) be a scale. A mapping g: A -> B is an admissible transformation iff (**A**, **B**, g ) is also a scale.

Real scales are classified according to the 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, a > 0 |
| Ratio Scale | g(x) = a x, a > 0 |
| Absolute Scale | g(x) = x |

**Figure 6.1:** Scale types of real scales. It is a hierarchy of scale types. The lowest one is the nominal scale and the highest one is the absolute scale.

**Meaningfulness**

However, the major question for the user is: how does he know what scale type is assumed and what are the conditions for the use of a measure on a certain scale level. Or equivalently, how does a measure and reality look like which creates numbers which can be transformed by a certain admissible transformation of scales.

Admissible transformations also lead to the definitions of meaningful statements /ROBE79/, p.58. *A statement with measurement values is meaningful iff its truth or falsity value is invariant to admissible transformations*. Meaningfulness guarantees that the truth value of statements with measurement values is invariant to scale transformations. For example if we say that the distance D1 is twice as long as distance D2, then this statement is true or false no matter whether length is measured in meters or yards. These problems are existing in the area of software measures too. This is the case if we want to make statements with measurement values for example after having applied statistical methods. We want to explain this problem by a statement which was given by Pressman /PRES92/ (Relation >) and similarly by Weyuker /WEYU88/.

**Example 6.1 (Wholeness)**

Let P1 and P2 be program bodies combined in some way to P1 o P2. Let u be a software complexity measure. Then the requirement for software complexity by Pressman is

$$u(P1 \text{ o } P2) > u(P1) + u(P2).$$

This statement is not meaningful for an interval scale. If we apply the admissible transformation of the interval scale g(x)=a x+b, then we get:

$$a\,u(P1 \circ P2) + b > a\,u(P1) + b + a\,u(P2) + b,$$

for all a>0 and for all b ε ℜ. Hence, the truth or falsity of the statement is not invariant to this type of admissible transformation. Hence the statement is meaningful for a ratio scale.

We see that meaningfulness depends on admissible transformations. The admissible transformations depend on the relational systems under consideration. Hence it is important to study the conditions which should hold on the relational systems in order to have a certain class of admissible transformations or equivalently to have a certain scale type.

In Bollmann and Zuse /BOLL93/ and Zuse /ZUSE94a/ we showed that wholeness is a pseudo property without any empirical meaning. We also showed that wholeness leads to non-additive ratio scales.

## 6.2 Ordinal and Ratio Scale

We now introduce the conditions for the use of software measures as an ordinal and a ratio scale. Firstly, we consider the conditions of the ordinal scale.

### 6.2.1 Ordinal Scale

In order to describe a measure as an ordinal scale we introduce the **weak order** which is a binary relation that is transitive and complete:

$P \bullet\geq P', P' \bullet\geq P'' \Rightarrow P \bullet\geq P''$ transitivity
$P \bullet\geq P'$ or $P' \bullet\geq P$ completeness (connectedness),
for all P, P', P'' ε **P**, where **P** is the set of flowgraphs ($\bullet\geq$ is a binary empirical relation, like *equal or more complex).*

In /ROBE79/, p.110, we find the following theorem which we can be applied directly to flowgraphs.

**Theorem 6.1:**
Suppose (**P**, $\bullet\geq$) is an empirical relational system, where **P** is a non-empty countable set of flowgraphs and where $\bullet\geq$ is a binary relation on **P**. Then there exists a function u: P -> ℜ, with

$$P \bullet\geq P' \Leftrightarrow u(P) \geq u(P')$$

for all P, P' ε **P**, iff $\bullet\geq$ is a weak order. If such a homomorphism exists, then

$$((\mathbf{P}, \bullet\geq), (\Re, \geq), \mathbf{u})$$

is an ordinal scale.

### 6.2.2 Ratio Scale

In order to come to the ratio scale the relational system (**P**, $\bullet\geq$) has to be extended to (**P**, $\bullet\geq$, o). We want to introduce the following notation for P, P'ε **P**: $P \approx P'$ iff $P \bullet\geq P'$, and $P' \bullet\geq P$, $P \bullet> P'$ iff $P \bullet\geq P'$, and not $P' \bullet\geq P$, where $\bullet\geq$ is the weak order and means equally or more complex, $\bullet>$ means more complex, and ≈ means equally complex.

**Theorem 6.2** /KRAN71/, p.74:
Let **P** be a non empty set, $\bullet\geq$ is a binary relation on **P**, and o a closed binary operation on **P**. Then (**P**, $\bullet\geq$, o) is a closed extensive structure iff there exists a real-valued function on P such that for all a, b ε **P**

$$(1)\ a \bullet\geq b \Leftrightarrow u(a) \geq u(b)$$
and
$$(2)\ u(a \circ b) = u(a) + u(b)$$

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

$$u'(a) = u(a).$$

The statement u' (a) = u(a) gives the admissible transformation for an additive ratio scale. We see that Theorem 6.2 gives us conditions for the additive ratio scale.

In Zuse /ZUSE91/, p.57 an extensive structure, as proposed by Bollmann /BOLL84/, is presented.

**Definition 6.2: (Extensive Structure):**
Let **P** be a non-empty set, $\bullet\geq$ binary relation on **P**, and o a closed binary operation on **P**. The relational system (**P**, $\bullet\geq$, o ) is an **extensive structure** if and only if the following axioms hold for all P1,.,P4 ε **P**.

A1': (**P**,$\bullet\geq$ ) is a weak order

A2': $P1 \circ (P2 \circ P3) \approx (P1 \circ P2) \circ P3$, axiom of weak associativity

A3': $P1 \circ P2 \approx P2 \circ P1$, 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
®

The axioms of the extensive structure describe empirical conditions related to concatenation operations of objects. The axiom of positivity: P o P' •> P, is also an axiom of the extensive structure.

### 6.3 Consequences of the Extensive Structure

We summarize the consequences of an extensive structure.

- An extensive structure is assumed by an additive Measure u.
- An extensive structure is also assumed by every Measure u', that is a strictly monotonic function f of an additive Measure u. It holds: u' = f u.
- An additive Measure u leads to the additive ratio scale.
- A Measure u', which is not additive, can be used as a non-additive ratio scale, if the combination rule is meaningful for the ratio scale.
- If the empirical conditions A1'-A5' are fulfilled in reality, then the Theorem of the extensive structure says, that an additive measure exists.
- Because the condition of the extensive structure are empirically, the extensive structure gives an qualitative interpretation of a measure.

### 6.4 Measurement Theory and Software Measures

We now show very briefly how measurement theory can be applied to software complexity measures. We will illustrate this with the Measures of McCabe. We specialize the general measurement theory approach for software complexity. We consider the empirical relational system (**P** , •≥) for the ordinal scale and (**P** , •≥, o) for the ratio scale, where **P** is the set of flowgraphs, •≥ the binary relation *equal or more complex than,* and o is a binary operation.

In measurement theory a concatenation operation is defined as:

$$A \times A \rightarrow A.$$

where A is the set of flowgraphs. As binary operations we consider here the sequential combination BSEQ for flowgraphs.

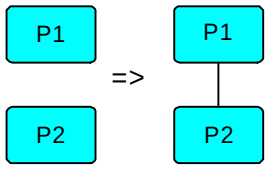


**Figure6.2:** The binary operation BSEQ=P1 o P2 of two arbitrary flowgraphs P1 and P2, and the sequential concatenation operation of a structured chart.

#### 6.4.1 Application of Measurement Theory to the Measure of McCabe

Firstly, we choose the Measures of McCabe /ZUSE91/, p.151, with

$$MCC\text{-}V=|E| - |N| + 2 \text{ , and}$$

$$MCC\text{-}V2=|E| - |N| + 1$$

to demonstrate our method. McCabe derived a software complexity measure from graph theory using the definition of the cyclomatic number. McCabe interpreted the cyclomatic number as the "minimum number of paths" in the flowgraph. He argued that the minimum number of paths determines the complexity (Called by McCabe: cyclomatic complexity) of the program: *The overall strategy will be to measure the complexity of a program by computing the number of linearly independent paths v(G), control the "size" of programs by setting an upper limit to v(G) (instead of using just physical size), and use the cyclomatic complexity as the basis for a testing methodology.*

The cyclomatic number is only a value of a mathematical function. Considering the Triple (**A**, **B, u**) for a scale, only the part (**B**,) is used, but measurement considers always the Triple (**A**, **B**, u) which includes the empirical relational system **A**. However, interpreting the cyclomatic number as the complexity of a flowgraph gives this number an empirical interpretation. Having an empirical interpretation of the numbers (cyclomatic complexity) the use of measurement theory is justified. We consider the empirical relational system (**P**, •≥) for the ordinal scale and (**P**, •≥ ,o) for the ratio scale, where **P** is the set of flowgraphs, o is the sequential combination BSEQ of flowgraphs as defined in Figure 6.1,

and MCC-V2: is the additive Measure MCC-V2=|E| - |N| + 1 of McCabe with

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

This combination rule is meaningful for the admissible transformation of a ratio scale.

#### 6.4.1.1 The Measure of McCabe as an Ordinal Scale

The Measure MCC-V2 can be used as an ordinal scale if the empirical ranking order corresponds with the ranking order of the Measure of McCabe. The Measure MCC-V=|E|-|N|+2 of McCabe is a strictly monotonic transformation of the Measure MCC-V2. A strictly monotonic function is the admissible transformation of the ordinal scale, it does not change the ranking order.

#### 6.4.1.2 The Measures of McCabe as a Ratio scale

In order to give the conditions for the use of the Measure MCC-V2 as a ratio scale we use Theorem 6.2. If o is the sequential combination BSEQ of flowgraphs, if complexity is transitive and complete and if the Measure MCC-V2 is an ordinal scale then

$$(\mathbf{P}, \bullet\geq, o)$$

is a closed extensive structure. In this case

$$((\mathbf{P}, \bullet\geq, o), (\Re, \geq, +), \text{MCC-V2}),$$

where MCC-V2=|E| - |N| + 1, is an additive ratio scale. The Measure MCC-V2 is a strictly monotonic transformation of: MCC-V. Put in other words: the Measure MCC-V2=|E| - |N| + 1can be used as a ratio scale because it is additive related to the concatenation operation BSEQ.

#### 6.4.1.3 The Measure of McCabe as an Absolute Scale

There is a confusing situation in literature about the absolute scale. It is not true that the absolute scale is described by counting. We now show the conditions for the use of the Measure of McCabe as an absolute scale. Very often people say that the Measure of McCabe measures the cyclomatic number and that is an absolute scale. That is wrong, because we can transform the cyclomatic number by an admissible transformation. We can say that we have twelve independent path, but we also can say that we have one dozen independent paths.

In order to characterize the Measure of McCabe as an absolute scale, we have to introduce an additional concatenation operation BALT. Above, we introduced the sequential concatenation operation BSEQ, additionally we introduce the concatenation operation BALT, which is defined as:

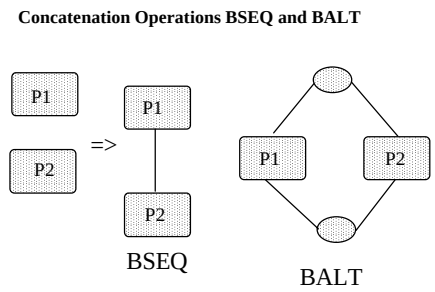


**Figure 6.3:** Concatenation operations BSEQ and BALT, where two Flowgraphs P1 and P2 are combined to a new Flowgraph BSEQ = P1 o P2 or BALT = P1 • P2, with P1, P2 ∈ **P.**

In order to come to the absolute scale we consider the following empirical relational system:

$$\mathbf{A} = (P, \bullet\geq, o, \bullet),$$

where P is the set of flowgraphs, •≥ an empirical relation, like equally or more difficult to maintain, o is the sequential concatenation operation BSEQ and • the concatenation operation BALT. We consider the Measure MCC-V2 of McCabe which is defined as:

$$\text{MCC-V2 } (P) = |E| - |N| + 1.$$

For the sequential concatenation operation BSEQ = P1 o P2 the combination rule is:

$$\text{MCC-V2 } (P1 \text{ o } P2) = \text{MCC-V2 } (P1) + \text{MCC-V2}(P2).$$

It is easy to see that the Measure MCC-V2 is additive related to the concatenation operation BSEQ. The combination rule for the concatenation operation BALT = P1 • P2 is:

$$\text{MCC-V2 } (P1 \bullet P2) = \text{MCC-V2}(P1) + \text{MCC-V2 } (P2) + 1.$$

We now consider whether both combination rules are meaningful related to admissible transformation the ratio scale. It is easy to see that the combination rule for the concatenation operation BSEQ is meaningful

related to the admissible transformation of the ratio scale. We illustrate this here:

$$a\ MCC\text{-}V2\ (P1\ o\ P2) = a\ MCC\text{-}V2\ (P1) + a\ MCC\text{-}V2(P2).$$

We transform each measurement value with the admissible transformation of the ratio scale (a>0). It is easy to see that the combination rule is invariant to the admissible transformation of the ratio scale.

We now consider the combination rule for the concatenation operation BALT and apply the admissible transformation of the ratio scale.:

$$a\ MCC\text{-}V2\ (P1 \bullet P2) = a\ MCC\text{-}V2(P1) + a\ MCC\text{-}V2\ (P2) + 1.$$

It is easy to see that the statement above is not invariant to the admissible transformation of the ratio scale (a>0). The statement above is invariant for a=1, and that is the admissible transformation of the absolute scale.

This shows that the Measure MCC-V2 can be used as an absolute scale if we consider the following empirical (**A**) and numerical (**B**) relational systems:

$$\mathbf{A} = (P, .\bullet\geq, o, \bullet), \text{ and}$$
$$\mathbf{B} = (R, \geq, +, MCC\text{-}V2(P1) + a\ MCC\text{-}V2\ (P2))$$

and the Measure MCC-V2 To use a measure as an absolute scale depends on the behavior of the concatenation operations related to the admissible transformations of the ratio scale. If one of the both combination rules are not invariant to the admissible transformation of the ratio scale, then we have an absolute scale (without any proof).

### 6.5 Concatenation Rules and Independence Conditions

Another important criteria for software measures are the independence conditions which are based on concatenation rules /ZUSE92/. Let us assume that **((A, •≥),(ℜ, ≥), u)** is an ordinal scale. We want to discuss the question whether there exists a binary operation • such that

$$u(P1\ o\ P2) = u(P1) \bullet u(P2)$$

for all P1, P2 ε A. We denote • as a combination rule. The formula above is identical with the formula

$$\mu(P1\ o\ P2) = f\ (\mu(P1), \mu(P2)).$$

The answer is given in the following theorems.

**Theorem C1:** There exists such a • iff P1≈ P2 => P1 o P ≈ P2 o P, and P1 ≈ P2 => P o P1 ≈ P o P2, for all P1, P2, P ε **P**

**Theorem C2:**
There exists such a • iff P1 ≈ P2 <=> P1 o P ≈ P2 o P, and P1 ≈ P2 <=> P o P1 ≈ P o P2, for all P1, P2, P ε **P**

**Theorem C3:** There exists such a • iff P1 •≥ P2 => P1 o P •≥ P2 o P, and P1 •≥ P2 => P o P1 •≥ P o P2, for all P1, P2, P ε **P**

**Theorem C4:** There exists such a • iff P1 •≥ P2 <=> P1 o P •≥ P2 o P, and P1 •≥ P2 <=> P o P1 •≥ P o P2, for all P1, P2, P ε **P**

The condition C1-C4 are hierarchical ordered, that means C4 implies C3, C2, and C3, C2 imply C1. The independence conditions are very important in the context of prediction and validation of measures. For more information see /ZUSE92/, and /ZUSE94a/.

### 6.6 Non-Additive Ratio Scales

Above, it was shown that the extensive structure leads to the additive ratio scale. The question is whether non-additive ratio scales are also possible. Remember, that wholeness was defined as:

$$u(P1\ o\ P2) > u(P1) + u(P2),$$

where P1, P2 ∈ **P**. We also showed that the statement of wholeness is meaningful for the admissible transformation of the ratio scale. The question is what type of Measure is assumed a non-additive combination rule which is meaningful for the non-additive ratio scale. Such a measure is of the type

$$u' = u^b,$$

with b>0. An additive Measure u has the following combination rule:

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

where P1, P2 ∈ **P**. It also holds $u = u'^{1/b}$. If we insert this function in the additive combination rule, we get:

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

with b>0. This combination rule is meaningful for the ratio scale. Put in other words: A Measure u' = $u^b$, where Measure u assumes an extensive structure and is additive, can be used as a non-additive ratio scale. The basic COCOMO-Model has such a combination rule. Taking the Measure MCC-V2= |E| - |N| + 1, which is additive related to the concatenation operation BSEQ, the Measure MCC-V2' = MCC-$V2^b$, with b>0, creates a non-additive ratio scale.

### 6.7 Validation and Prediction

If we have two ratio scales, one of the measure M and the other one of the external variable V, then the only one function which is possible between two ratio scales is the following formula:

$$V(P) = a\ M^b.$$

with a,b>0. This formula has to be validated if we want to use a measure, which assumes a ratio scale, as a predictor for an external variable C, which can be also used as a ratio scale. The formula above is known as the basic COCOMO-Model /BOEH81/. For more information's see Bollmann et al. /BOLL93/ and Zuse /ZUSE94a/.

## 7 References


/BASI84/ Basili, V.; Perricone, Barry T.: Software Errors and Complexity: An Empirical Investigation. Communications of the ACM, Volume 27, No. 1, January 1984, pp. 42-52.

/BOLL93/ Bollmann-Sdorra, P.; Zuse, H.: Prediction Models and Software Complexity Measures from a Measurement Theoretic View. Proceedings of the 3rd International Software Quality Conference, Lake Tahoe, Nevada, October 4-7, 1993.

/BOOC91/ Booch, G.: Object-Oriented Design with Applications. Benjamin/Cummings, 1991.

/BUER87/ Bürger, Hans-Christopf: Axiomatische Beschreibung von Glaubensfunktionen. Diploma Thesis, TU-Berlin, Dezember 1987.

/CHID94/ Chidamber, Shyam, R.; Kemerer, Chris, F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, Volume 20, No. 6, June 1994, pp. 476-493.

/CHUR95/ Churcher, Neville, I.; Shepperd, Martin, J.: Comments on "A Metrics Suite for Object Oriented Design". IEEE Transactions on Software Engineering, Vol. 21, No. 3, March 1995, pp. 1-3.

/FETC95/ Fetcke, Thomas: Software Metriken bei der Object-orientierten Programmierung. Diploma thesis, Gesellschaft für Mathematik und Datenverarbeitung (GMD), St. Augustin, and TU-Berlin, 1995.

/KRAN71/ Krantz, David H.; Luce, R. Duncan; Suppes; Patrick; Tversky, Amos: Foundations of Measurement - Additive and Polynominal Representation, Academic Press, Volume 1, 1971

/LUCE90/ Luce, R. Duncan; Krantz, David H.; Suppes; Patrick; Tversky, Amos: Foundations of Measurement. Volume 3, Academic Press, 1990

/ROBE79/ Roberts, Fred S.: Measurement Theory with Applications to Decisionmaking, Utility, and the Social Sciences. Encyclopedia of Mathematics and its Applications Addison Wesley Publishing Company, 1979.

/SHAF76/ Shafer, Glenn: A Mathematical Theory of Evidence. Princeton University Press, 1976.

/SHAF87/ Shafer, Glenn: Belief Functions and Possibility Measures. In Analysis of Fuzzy Information, Volume 1, Mathematics and Logic, J.C. Bezdek, Ed. Boca Raton, FL, CRC Press, 1987, pp. 51-84.

/WEYU88/ Weyuker, Elaine J.: Evaluating Software Complexity Measures. IEEE Transactions of Software Engineering Volume 14, No. 9, Sept. 88.

/WONG91/ Wong, S., K., M.; Yao, Y., Y.; Bollmann-Sdorra, P.; Bürger, H.,C.: Axiomatization of Qunatitative Belief Structure. IEEE Transaction on Systems, Man and Cybernatics, Volume 21, No. 4, July/August 1991.

/ZUSE89/ Zuse, Horst; Bollmann, P.: Using Measurement Theory to Describe the Properties and Scales of Static Software Complexity Metrics. SIGPLAN Notices, Volume 24, No. 8, pp.23-33, August 89.

/ZUSE91/ Zuse, Horst: Software Complexity: Measures and Methods. DeGruyter Publisher 1991, Berlin, New York, 605 pages, 498 figures.

/ZUSE92/ Zuse, Horst; Bollmann-Sdorra, Peter: Measurement Theory and Software Measures. In: Workshops in Computing: T.Denvir, R.Herman and R.Whitty (Eds.): Proceedings of the BCS-FACS Workshop on Formal Aspects of Measurement, South Bank University, London, May 5, 1991. Series Edited by Professor C.J. Rijsbergen. ISBN 3-540-19788-5. Springer Verlag London Ltd, Springer House, 8 Alexandra Road, Wimbledon, London SW19 7JZ, UK, 1992.

/ZUSE94a/ Zuse, Horst: Software Complexity Metrics/Analysis. Marciniak, John, J. (Editor-in-Chief): Encyclopedia of Software Engineering, Volume I, John Wiley & Sons, Inc. 1994, pp. 131-166.

/ZUSE94b/ Zuse, Horst: Software Measurement - An Overview. Technical Paper, TU-Berlin, FR5-3, 1994.

/ZUSE94c/ Zuse, Horst: Foundations of the Validation of Object-Oriented Software Measures. In: Theorie und Praxis der Softwaremessung (Dumke, R.; Zuse, H. (Editors), Deutsche Universitätsverlag DUV, Gabler - Vieweg - Westdeutscher Verlag, 1994, pp. 136-214.


### About the Author:


**Horst Zuse** received his B.S. degree in electrical engineering from the Technische Universität of Berlin, Germany, in 1970, the Diploma degree in electrical engineering from the Technische Universität in 1973, and the Ph.D. Degree in computer science from the Technische Universität of Berlin in 1985. Since 1975 he is senior research scientist with the Technische Universität Berlin. His research interests are information retrieval systems, software engineering, software metrics and the measurement of "complexity"

and "quality" of software during the software life-cycle. From 1987 to 1988 he was for one year with IBM Thomas J. Watson Research in Yorktown Heights. His research work there was software metrics, too. In 1991 he published the book: Software Complexity - Measures and Methods (De Gruyter Publisher). From 1989 till 1992 he was with the ESPRIT II Project 2384 METKIT (Metric-Educational- Toolkit) of the European Commission. Since 1990 he gives several times a year seminars about software metrics for people of the industry within the scope of DECollege and ORACLE. He also gave many presentations and seminars about software measurement on conferences in US and Canada. Now, his research interests are validation of software metrics, evaluation of cost estimation and prediction models, application of software metrics in the whole software life-cycle and standardization of software measures. From August 1, 1994 to December 31, 1994 he is invited as a researcher by the *Gesellschaft für Mathematik und Datenverarbeitung (GMD)* in St. Augustin in Germany.