International Software Engineering Research Network technical report #ISERN-95-03

Theoretical and Empirical Validation of Software Product Measures[1]

Lionel Briand
Centre de Recherche Informatique de Montréal (CRIM)
Software Engineering Group
1801 McGill College av.
Montréal, PQ, H3A 2H4
Canada
e-mail: lbriand@crim.ca

Khaled El Emam
Centre de Recherche Informatique de Montréal (CRIM)
Software Engineering Group
1801 McGill College av.
Montréal, PQ, H3A 2H4
Canada
e-mail: kelemam@crim.ca

Sandro Morasca
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza L. Da Vinci 32,
I-20133, Milano
Italy
e-mail: morasca@elet.polimi.it

Abstract

In this paper we present a concrete method for validating software product measures for internal attributes and provide guidelines for its application. This method integrates much of the relevant previous work, such as measurement theory, properties of measures, and GQM. We identify two types of validation: theoretical and empirical. The former addresses the question "is the measure measuring the attribute it is purporting to measure?", and the latter addresses the question "is the measure useful in the sense that it is related to other variables in expected ways?"

1. Introduction

Recent software engineering literature has reflected a concern for methods to validate measures for internal software attributes (e.g., see [S92][FK90]). This concern is driven, at least partially, by a recognition that: (i) common practices for validation are not acceptable, and (ii) valid measures are essential for software project management and sound empirical research. It is therefore crucial that the software engineering community reach consensus on precise methods for validating measures.

In this paper we present a concrete method for validating measures of internal software product attributes and provide guidelines for its application. This method covers two types of validation: theoretical and empirical. We also integrate many of the concepts that are relevant for validation into this method (e.g., measurement theory [F91][Z91], properties of measures [BMB(b)94], and GQM [BR88]).

The paper is organized as follows. In the next section we present our definition of validity. In Section 3 we present how to theoretically validate a measure. In Section 4 we present how to empirically validate a measure. We conclude the paper in Section 5 with an overall summary of the paper.

[1] This paper appears as Technical Report number ISERN-95-03, International Software Engineering Research Network, 1995.

2. A Definition of Validity

Fenton [F91] identifies three classes of entities that are of interest in software engineering measurement: (a) products, (b) processes, and (c) resources. What one would measure are the attributes of entities belonging to these three classes. For instance, the complexity (attribute) of code (a product entity). Our concern in this paper is with the measurement of the internal attributes of product entities. An internal attribute, as defined in [F91], can be measured purely in terms of the product itself and not with respect to how it relates to its environment.

Such attributes are interesting for software engineers as long as they are a part of software engineering theories. A theory defines relationships amongst product, process, and resource attributes. For example, a simple theory could define a negative association between code complexity and maintainability. As another example, consider Parnas' theory about design [P72], which is commonly accepted among researchers and practitioners. Parnas' theory states that high cohesion within modules and low coupling across modules are desirable design attributes, in that a software system designed accordingly is easier to understand, modify, and maintain. When supported by convincing empirical evidence, theories help us better understand software engineering phenomena and can also be useful for prediction purposes. Thus, if an attribute is not part of any theory, one must question whether the attribute is worth studying at all.

It is quite surprising that, in an empirical discipline such as software engineering (i.e., no one can mathematically prove the effectiveness of technologies), theories are often accepted without any empirical proof of their truthfulness. This is not at all to say that theories like Parnas' should be discarded because no empirical evidence is provided when they are proposed. However, we point out the need for thorough theoretical and empirical study before these theories get accepted.[2] Theories cannot be empirically tested unless we can measure the attributes involved in these theories. Therefore, for example, we cannot study the relationship between code complexity and maintainability unless we can measure the attributes of complexity and maintainability.

Two approaches for validation have been prescribed and/or practiced in software engineering: (a) theoretical validation (e.g., see [FK90]), and (b) empirical validation (e.g., see [S92]). These two types of validation are used to demonstrate respectively that: (a) a measure is really measuring the attribute it is purporting to measure, and (b) the measure is useful in the sense that it is related to other variables in expected ways (as defined in the theories).

To demonstrate that a measure is really measuring the attribute it is purporting to measure is the most basic form of validation. It is also a prerequisite to demonstrating its usefulness. Obviously, one cannot properly test theories if one is not sure that s/he is measuring the attributes in those

---

[2] As a matter of fact, in [BMB94(a)] we have provided such a study of Parnas' design principles. The result of the study basically confirms Parnas' ideas. However, 22 years elapsed between Parnas' proposal and this study!

theories. On the other hand, as mentioned earlier, it is questionable whether it is worth showing that a measure is measuring a particular attribute if that attribute is not part of a theory.

Therefore, we will say that a measure is valid if

1. it actually measures what it purports to measure,
2. it is useful, i.e., it is related to some external attribute worth studying and therefore helps reach some goal (e.g., assessment, prediction).

In the following two sections we describe the processes for the theoretical and empirical validation of internal product attributes. These processes are an attempt at reconciling the different views regarding validation of measures that have appeared in the literature, and to integrate validation more closely with some of the elements of measurement theory.

3. Theoretical Validation

3.1 Defining Empirical Relational Systems

Theoretical validation is concerned with demonstrating that a measure is measuring the concept it is purporting to measure. The first requirement for theoretical validation is that either the analyst has a good intuitive understanding of the concept that is being measured and/or that the software engineering community has a consensual intuitive understanding of the concept. This means that a measure μ of an attribute must be consistent with the intuitive understanding of this attribute. For instance, if μ is supposed to measure the "height" of people, and we empirically know that person P1 is taller than person P2, we require that μ(P1) > μ(P2).

Theoretical validation then involves modeling this intuitive understanding of the attribute we want to measure. In the framework of measurement theory [F91, Z91], intuition and empirical knowledge are modeled by empirical relational systems, whose definition is provided below. (This definition and all definition related to the basics of measurement theory are taken from [Z91, pp. 40 - 51], based on [R79].)

A relational system **A** *is an ordered tuple* $(A,R_1,\ldots,R_n,o_1,\ldots o_m)$ *where A is a nonempty set of objects, the* R_i*,* $i = 1,\ldots,n$ *are ki-ary relations on A and the* o_j*,* $j=1,\ldots,m$ *are closed binary operations.*

Empirical Relational System:

A $= (A,R_1,\ldots,R_n,o_1,\ldots o_m)$.

A *is a non-empty set of empirical objects that are to be measured (in our case program texts, flowgraphs, etc.).*

R_i *are ki-ary empirical relations on A with* $i = 1,\ldots,n$*. For example, the empirical relation "equal to or more complex".*

o_j are binary operations on the empirical objects A that are to be measured (for example a concatenation of control flowgraphs) with j=1,…,m.

The empirical relational system describes the part of reality on which measurement is carried out (via the set of objects A) and our empirical knowledge of the objects' attributes we want to measure (via the collection of empirical relations R_i's).

Depending on the attributes we want to measure, different relations are used. For instance, if we are interested in program length, we may want to use the relation "longer than" (e.g., "program P_1 is longer than program P_2"); if we are interested in program complexity, we may want to use the relation "more complex than" (e.g., "program P_3 is more complex than program P_4"). Binary operations may be seen as a special case of ternary relations between objects. For instance, suppose that o_1 is the concatenation operation between two programs. We may see it as a relation Concat(Program1, Program2, Program3), where Program3 is obtained as the concatenation of Program1 and Program2, i.e., Program3 = Program1 o_1 Program2. It is important to notice that an empirical relational system does not contain any reference to measures or numbers. Only "qualitative" statements are made, based on our intuition and knowledge of the attribute.

Usually, the relations contained in empirical relational systems are of different kinds. We will provide here three instances. We will assume that we want to build an empirical relational system for the size of program bodies, i.e., executable sections of programs, as defined in [W88].

A first set of relations defines partial orders. For instance, for any triplet (Program1, Program2, Program3) as defined above, the following empirical relational system property holds:

(Concat(Program1, Program2, Program3) ⇒ Program3 » Program1) **and**
(Concat(Program1, Program2, Program3) ⇒ Program3 » Program2)

where “»” is an empirical relation meaning “has a larger or equal size than.” In this example, such a property defines a partial order on the set of program bodies.

Equivalence relations constitute another type of usually defined relations. For instance, for any quadruple of program bodies (Program1, Program2, Program3, Program4), the following empirical relational system property holds:

(Concat(Program1, Program2, Program3) and
Concat(Program2, Program1, Program4)) ⇒ Program3 ~ Program4

where “~” is an empirical equivalence relation meaning “has the same size as.”

A last example of relation is the identification of special objects. For instance, one may assume that there is a particular program body, the empty program body, whose size is less than the size of any other program body. Notice that this alone does not imply that the empty program body should have size equal to zero, since no measure has been defined yet.

One's understanding of the empirical relational system of the attribute s/he is trying to measure is crucial in deriving valid and useful measures. However, confusion in terminology and lack of consensus in the software engineering community make it more difficult to formalize empirical relational systems. "Complexity" is a good example of confusion in software engineering [BMB94(b)]. We therefore believe it is important that software measurement researchers agree on a basic set of properties that the empirical relational systems of common internal attributes (i.e., complexity, coupling, etc.) should have. This will help software engineering researchers and practitioners (1) to use a less ambiguous terminology and (2) have guidelines to define their own empirical relational systems in order to derive useful measures.

In addition, we also believe that the building of an empirical relational system should be based on a measurement goal. This stems from the fact that measurement goals will determine theories to be built (e.g., relationship between data flow complexity and maintainability) and external attributes of interest (e.g., modifiability, portability). Therefore, they will provide a context to the definition of empirical relational systems and to the derivation of measures of internal attributes. Thereby, in a context precisely defined by measurement goals, empirical relational systems of attributes can be made more precise even though the validity of their definition is *a priori* limited to the context in which they were defined. Besides, one should always keep in mind that measurement is an expensive activity, which should be targeted at the achievement of business goals relevant to the organization in which it takes place. There is little point in measuring without a clearly defined context and goals: that is not likely to yield any useful and sensible results.

More precisely, the following factors influence the building of an empirical relational system:

1. **Object of study.** Different objects of study (e.g., design vs. code) constitute the entities of different empirical relational systems. As a consequence, the empirical relations between them might be different.
2. **Quality focus.** Different quality attributes of the entities may be of interest (i.e., modifiability vs. error-proneness). According to Fenton’s classification of attributes (i.e., internal vs. external), these quality attributes are external attributes of the entities. Based on the external attribute(s) of interest, a set of relevant entities and internal attributes is identified. The assumption is that the identified internal attributes are related to the external attributes of interest in some predetermined way. We concentrate on a well identified set of attributes that we believe relevant, instead of studying a broad set of attributes selected without clear justification, many of which may have no clear link to the phenomenon we want to study. The exploratory study of a large set of attributes, and, consequently, the definition of a large number of metrics, may have dangerous consequences [CG93]. For instance, starting from an attribute which is not relevant to the goal, one may define a

measure which turns out to be statistically well correlated to the external attribute of interest only by chance. The result obtained is of difficult and uncertain interpretation, and may lead to incorrect decisions.

3. **Environment in which the goal is defined.** The empirical relational system embodies intuition and knowledge that are specific to the people that belong to some environment, in which the defined goal is relevant. Therefore, the empirical relational system is valid in that environment and may not be valid elsewhere. One should always be very careful when reusing empirical relational systems (and the measures derived from them) across environments.
4. **Viewpoint.** Depending on people's goals and role (e.g., design vs. implementation) in a software organization, different sets of entities (e.g., design vs. code artifacts) and different internal attributes may be selected (e.g., component coupling vs. control flow complexity). Once again, we emphasize that empirical relational systems are inherently subjective.

The four factors mentioned above are four of the dimensions of GQM goals [B92, BR88]. Here is a summary of a template that may be used to define goals:

Object of study*:* products, processes, resources
Purpose*:* characterization, evaluation, prediction, improvement, ...
Quality focus*:* cost, correctness, defect removal, changes, reliability, ...
Viewpoint*:* user, customer, manager, developer, corporation, ...
Environment: people, resources, processes, …

The following is an example of a GQM goal:

Object of study: C++ Classes
Purpose: prediction
Quality focus: likelihood of defect detection
Viewpoint: designer
Environment of Study: company X, OO systems developed in C++ in a given application domain

The GQM paradigm can be used even further as described in the references mentioned above. Starting from a goal, questions will be used to identify the internal attributes of interest and better characterize the external attribute. Based on these questions, suitable measures are derived for both internal and external attributes of interest. For instance, questions may be asked about the complexity and size (internal attributes) of the studied C++ classes, and about what is considered a defect (external attribute)—different environments have different definitions for defects. Measures should then be defined for complexity and size of C++ classes and for the number and criticality of uncovered defects.

As a side remark, *Purpose* is the only goal dimension that does not influence the building of an empirical relational system. However, it will influence the empirical validation, as we will show in Section 4.

3.2 Defining Formal Relational Systems

Once an empirical relational system is defined, a measure—as described in Definition 3.1 below—for the attribute of study can be defined. This is equivalent to saying that each entity in the set of entities of the empirical relational system is associated with a value. These values have to satisfy relations that preserve the relations existing between the entities in the empirical relational system—see Definition 3.2 below. Therefore, the empirical relations are mapped onto relations between the values of measures. The pair (values of a measure, relations between the values of the measure) is called a formal relational system.

Formal Relational System:

B $= (B, S_1, \ldots, S_n, \bullet_1, \ldots \bullet_m)$.

B *is a non-empty set of formal objects, for example numbers or vectors.*

Si *are ki-ary relations on B such as "greater than" or "equal to or greater than".*

$\bullet_j$ *are closed binary operations B such as addition or multiplication.*

Every object *a* of A is mapped into a value of B, i.e., it is measured according to measure $\mu(a)$. Every empirical relation R_i is mapped into a formal relation S_i. For instance, the relation "more complex than" between two programs is mapped into the relation ">" between the complexity measures of two programs. The formal relations must preserve the meaning of the empirical statements. For instance, suppose that R_1 is the empirical relation "more complex than," S_1 is the formal relation ">," and μ is a complexity measure. Then, we must have that program P_1 is more complex than program P_2 if and only if $\mu(P_1) > \mu(P_2)$.

Definition 3.1 *(Measure* μ*):*

A ***measure*** μ *is a mapping* $\mu: A \rightarrow B$ *which yields for every empirical object* $a \in A$ *a formal object (measurement value)* $\mu(a) \in B$*. Of course, this mapping may not be arbitrary. This leads to the following definition of a scale.*

Definition 3.2 *(Scale):*[3]

Let $\mathbf{A} = (A, R_1, \ldots, R_n, o_1, \ldots, o_m)$ *be an empirical relational system and* $\mathbf{B} = (B, S_1, \ldots, S_n, \bullet_1, \ldots \bullet_m)$ *a formal relational system and* μ *a measure. The Triple* $(\mathbf{A}, \mathbf{B}, \mu)$ *is a scale if and only if for all i,j and for all* $a_1, \ldots a_k, b, c \in A$ *the following holds*

$$R_i(a_1, \ldots a_k) \Leftrightarrow S_i(\mu(a_1), \ldots, \mu(a_k)) \text{ and } \mu(b\ o_j\ c) = \mu(b) \bullet_j \mu(c)$$

If B = R is the set of real numbers, the Triple $(\mathbf{A}, \mathbf{B}, \mu)$ *is a* ***real scale***.

The formal relational system describes (via the set B) the value domains of the measures for the studied objects' attributes. For instance, these may be integer numbers, real numbers, vectors of integer and/or real numbers, etc. A formal relational system also describes (via the collection of relations S_i's) the relations of interest between the measures.

The relations of the empirical relational system are translated into relations of the formal relational system. Typical kinds of relations are

1. **Inequalities.** Partial orders (see Section 3.1) on sets of entities may be translated into inequalities between measures.
2. **Equalities.** Equivalence relations may be translated into equalities.
3. **Assignment of special values.** For instance, the measure of the size of the empty program body may be set to the value zero

The measures we define are often intended to measure some standard attribute of entities, such as size, complexity, cohesion, coupling. We may want to check if this is true; for instance, we may want to check whether a measure we defined to measure size of program bodies is really a size measure. This can be done by using a set of properties which characterize all measures of a specified attribute. Instances of such properties can be found in the literature [W88, L91,TZ92, BMB94(b)]. Again, such properties are defined based on one's intuition. However, as mentioned above, we believe that some kind of generalized consensus must be reached on them, so as to

1. Avoid communication problems among researchers and practitioners.
2. Avoid definitions of ambiguous measures, whose meaning is uncertain.
3. Provide the modeler with an additional means to check whether his/her empirical modeling is correct. For instance, suppose that one believes that data flow complexity of software is an internal product attribute that is relevant to some specified goal. Suppose also that s/he has an intuitive

[3] Definition 4.2 was slightly changed as compared to the one in [Z91].

understanding of what a complexity measure should look like. After defining an empirical relational system (e.g., under the form of mathematical properties as explained in Section 3.1), a formal relational system, and a measure from the former to the latter, s/he may find out that the measure does not satisfy his/her intuitive understanding of what a complexity measure is. At this point, s/he may choose to either: revise the empirical relational system s/he defined, or accept that measure as a function that measures something other than data flow complexity.

3.3 Summary

To conclude this section, theoretical validation requires that:

1 an empirical relational system be defined, e.g., through the definition of properties and based on defined relations and operations between entities,
2 a suitable formal relational system be defined,
3 the properties of the empirical relational system be preserved by the formal relational system when the measure to be validated is used to do the mapping between them.

Such a validation is only meaningful within the predefined context described, for example, by GQM goals. Any change in a measurement goal might trigger changes in relational systems and therefore in the derived measures. In addition, we believe that basic properties for the definition of empirical relational systems for measurement concepts such as complexity, coupling, etc., should be the object of a consensus in the software engineering community. We propose such a set of basic and fundamental properties in [BMB(b)94].

As mentioned above, in this paper, we only focus on internal product attributes. In our discussion, we have assumed that an external attribute exists to which the internal measure is linked. However, the external attribute should undergo the same process as that used for the theoretical validation of an internal product measure. In other words, one should start from the measurement goal and define an empirical relational system for the external attribute of interest. It is generally easier to find sensible measures of external attributes than it is for internal ones. As an example, measures for cost and time (typical quality foci for process related goals), or for reliability and defects (typical quality foci for product related goals) are quite straightforward.

Once a measure can be considered valid from a theoretical point of view, the assumptions on which the empirical relational system is based and the measures must be validated empirically since no theoretical model can guarantee their truthfulness.

4. Empirical Validation

4.1 Basic Method for Empirical Validation

When one empirically validates a measure, then one is attempting to answer the following question: is the measure useful in a particular development environment, considering a given purpose in defining the measure and a modeler's viewpoint? In a given context, we believe the measure of a particular internal attribute is useful if it is related to a measure of some external attribute of the object of study, which is defined in the GQM context as the Quality focus, e.g., likelihood of defect detection. Measures of internal product attributes in software engineering are artificial concepts and do not hold any meaning in themselves. For example, a complexity measure, implicitly or explicitly, is always defined in relation to some external attribute such as error-proneness, effort, etc. In itself, a measure of the internal attribute of complexity, such as cyclomatic complexity, does not have any concrete meaning and usefulness besides its intuitive relationships with external attribute measures, e.g., likelihood of defect detection.

We want to remark that, to some extent, the same is true for other measures too. For example, the measures of size of an object have always, implicitly or explicitly, been defined in relation to some goal. For instance, consider two size measures that have been defined for objects, weight and volume. These measures do not have a concrete meaning in themselves, but only in relation to some goal (e.g., volume: "Will this object fit in the trunk of my car?"; weight: "Will my car be able to carry this object without breaking?"). These two measures now seem so natural and obvious, and seem to have a meaning in themselves. This is due to the fact that

1. the empirical relational system on which they are based captures well our understanding about the size attribute;
2. the size attribute is deemed to be relevant—in these two different "flavors"—to a variety of goals;
3. the two measures have been found useful in a large variety of circumstances, i.e., with respect to solving a set of practical problems.

Therefore, relating an internal attribute measure to an external attribute measure is a crucial point for the validation of measures. This requires the following three steps:

(1) Collection of data about the quality focus measure and the internal product measures of interest, e.g., complexity measures, defect instances of C++ classes. This will allow us, based on data analysis, to determine some optimal statistical relationship(s).

(2) Identification of the measurement level of internal and external attribute measures. This will determine, to some extent, the types of data analysis to be performed.

(3) Selection and use of suitable analysis and modeling mechanisms to formalize and quantify the relationship(s).

The three steps above are influenced by the *purpose* dimension of the GQM measurement goal. In particular, the purpose affects

- *type of data to be collected*, e.g., process improvement requires additional data over process prediction (e.g., with respect to development effort), in order to allow for the determination of optimal techniques and methods. For example, performance data are needed in sufficient amount to ensure a minimal level of confidence in the improvement decisions.
- *amount of data to be collected*, e.g., prediction usually requires more data than characterization so that the identified relationships/patterns are statistically significant. Characterization only requires that the data be representative of what is to be characterized.

Step (1) implies that we have to collect data that are representative of the environment of study, e.g., the physical organization of development, the application domain, the platform, the programming language. Unfortunately, there is no easy way to determine what piece of data is relevant or not in the context of a study. For further discussion, refer to [BBT94]. Step (2) and Step (3) are more complex issues than they appear and are the subjects of the next two subsections, respectively.

4.2 Level of Measurement and Statistical Techniques

In order to determine whether an internal product attribute is related to an external product attribute, one commonly has to select a statistical technique. Several books and papers on the topic of measurement theory are conveying the idea that scale types should be used to proscribe the use of "inappropriate" statistical techniques. For example, a table similar to the one shown in Figure 1 is given in [F91]. This table, for instance, proscribes the use of the Pearson product moment correlation for scale types that are either nominal or ordinal. Such proscriptions, of course, are not unique to software engineering. For instance, they were originally presented by the psychologist Stevens [Ste46], serve as the basis of the classic text of Siegel on nonparametric statistics [SC88], and serve as an integral part of the decision tree developed by Andrews et al. [AKD+81] to guide researchers in the selection of the most appropriate statistics. Accordingly, if a researcher's measures do not reach the interval level, it is advised that s/he use non-parametric statistics (i.e., tests which make less stringent assumptions).

Scale Type	**Examples of Appropriate Statistics**	**Type of Appropriate Statistics**
Nominal	Mode Frequency Contingency Coefficient	Nonparametric Statistics
Ordinal	Median Kendall's tau Spearman's rho	
Interval	Mean Pearson's correlation	Nonparametric and Parametric Statistics
Ratio	Geometric Mean Coefficient of Variation	

Figure 1: Appropriate statistics for various scale types.

However, in order to select the most “appropriate” statistics, a researcher has to know the type of scale(s) that s/he is using. The problem is that, in sofware engineering, like in other scientific disciplines, often it is very difficult to *determine* the scale type of a measure. For example, what is the scale type of cyclomatic complexity? Can we assume that the distances on the cyclomatic complexity scale are preserved across all of the scale? This is difficult to say and the answer can only be based on intuition. Despite a few available techniques to help the researchers in particular situations (see [BEM95]), the answer to those questions is hardly ever straightforward.

Therefore, there are many cases where researchers cannot demonstrate that their scales are interval, but they are confident that they are more than only ordinal. By treating them as ordinal, researchers would be discarding a good deal of information. Therefore, as Tukey [Tuk86a] notes *“The question must be ‘If a scale is not an interval scale, must it be merely ordinal?’”*

Is it realistic to answer questions about scale type with absolute certainty, since their answers always rely on intuition and are therefore subjective? Can we know for sure the scale types of the measures we use? Knowing the scale type of a measure with absolute certainty is out of the question in the vast majority of cases. And in those cases, should we just discard our practical questions—whose answers may have a real impact on the software process—because we are not 100% positive about the scale types of the measures we are using? To paraphrase Tukey [Tuk86b], "Science is not mathematics" and we are not looking for perfection and absolute proofs but for *evidence* that our theories match reality as closely as possible. The other alternative, i.e., reject approximate theories, would have catastrophic consequences on most sciences, and in particular, on software engineering. What is not acceptable from a strictly mathematical perspective may be acceptable evidence and even a necessary one from an engineering or an experimental perspective.

It is informative to note that much of the recent progress in the social sciences would not have been possible if the use of "approximate" measurement scales had been strictly proscribed. For example, Tukey [Tuk86a] states after summarizing Stevens' proscriptions *"This view thus summarized is a dangerous one. If generally adopted it would not only lead to inefficient analysis of data, but it would also lead to failure to give any answer at all to questions whose answers are perfectly good, though slightly approximate. All this loss for essentially no gain."* Similarly, in the context of multiple regression, Cohen and Cohen [CC83] state: *"The issue of the level of scaling and measurement precision required of quantitative variables in [Multiple Regression/Correlation] is complex and controversial. We take the position that, in practice, almost anything goes. Formally, fixed model regression analysis demands that the quantitative independent variables be scaled at truly equal intervals ... Meeting this demand would rule out the use of all psychological tests, sociological indices, rating scales, and interview responses ... this eliminates virtually all kinds of quantitative variables on which the behavioral sciences depend."* Even Stevens himself, with respect to ordinal scales, concedes that [Ste46]: *"In the strictest propriety the ordinary statistics involving means and standard deviations ought not to be used with these scales, for these statistics imply a knowledge of something more than relative rank-order of data. On the other hand, for this 'illegal' statisticizing there can be invoked a kind of pragmatic sanction: In numerous instances it leads to fruitful results."*

The above, evidently more pragmatic, view is under-represented in software engineering, however. This stems from the fact that some of the most influential books in our field (the ones considered as standard references on measurement theory in software engineering such as [F91][Z91]) are only presenting one side of the debate, (i.e., the side claiming that scale types should be used to proscribe data analysis techniques).

In most cases, the questions to answer (i.e., our measurement goals) determine the scale under which data must be used, and not *vice versa*. One should use the appropriate statistical technique assuming the level of measurement required by the question. If a pattern is detected, then the scientist should start thinking about the validity of the assumption s/he made about the scale types. In addition, it is sometimes possible to use different statistics assuming different scale types and compare the results.

We usually come up with an important question first (e.g., "Is there a linear relationship between two variables?"), relevant to our measurement goals. Subsequently, we usually look around for available data or develop measurement scales that we think can help us answer our questions at a reasonable cost. Also, most of the time, our learning process is exploratory and we have a very limited understanding of the phenomena we are studying, e.g., the impact of class coupling on defect detection likelihood. Some important questions require interval or ratio scales but we are not sure if the scales we are using are actually interval or ratio. Should we not analyze the data? For instance, if someone is asking the question: "If I can reduce coupling by 10% through a better design technique, how much would I gain in terms of reduction of defect density?" The answer to this—quite common—kind of question requires a ratio scale for coupling, since the reduction is given in terms of proportions (i.e., ratios), and there is a natural zero level of coupling (when modules are not related to each other). Intuitively, we can be quite sure defect density is defined on a ratio scale too. However, with respect to coupling, the level of measurement of the scale is quite difficult to determine, regardless of the definition used.

For example, if a statistically significant linear relationship is found between coupling and defect density through linear regression then, theoretically, the researcher must start wondering if the computed level of significance is real (or close to reality) since there is some uncertainty with respect to the type of the coupling scale. External information may be examined in order to confirm or otherwise the scale assumption. For example, assuming we want to model defect density, programmers may be surveyed by asking them to score the relative "difficulty" of programs with different coupling levels. If the scores confirm that the distance is, on average, preserved along the studied part of the scale (hopefully, the relevant one for the environment under study), then the equal interval properties may be assumed with greater confidence. In addition, thorough experience and a good intuitive understanding of the phenomenon under study can help a great deal. For example, in a given environment, very often programmers know the common causes of errors and their relative impact. Scales may thus be validated with the help of experts.

Such an approach is supported by numerous studies (for a more detailed discussion, see [BEM95]) which show that, in general, parametric statistics are robust when scales are not too far from being interval. In other words, when a scale is not an exponential distortion of an interval scale, the likelihood of error of type I (i.e., the null hypothesis is incorrectly rejected) does not significantly increase. In addition, other studies have shown that, in most cases, non-parametric statisics were of lesser power (i.e., higher likelihood of error of type II, i.e., the null hypothesis is falsely not rejected) than parametric statistics when their underlying assumptions were not violated to an extreme extent. Moreover, dealing with variable interactions in a multivariate tends to be much easier when using parametric techniques, e.g., multivariate regression analysis with interaction terms [DG84][AW91].

4.3 **Data Analysis and Modeling**

In order to identify relationships between the quality focus measure and the internal attribute measure to be validated, a number of statistical techniques for data analysis are available. When these two measures can be considered to be nearly or exactly at the interval-level of measurement, least-square regression analysis is commonly used. However, the specific functional form of the least-square regression equation is a matter of assumption and must be derived from prior knowledge of the modeler. It is always simpler and more convenient to deal with linear models and therefore, when possible, transformations may be applied to linearize the relationship between two measures.

For example, a very common transformation used in software engineering is the logarithmic transformation. This is applied frequently in the construction of effort estimation models using linear regression. As has been noted by Bailey and Basili [BB81] and Basili [Bas80], a general form of such models is:

$E = a\, L^{b}$
where:

E = effort
L = some measure of size (usually LOC)
a, b = constants

In this particular case, an analyst could use the following estimating equation and thus use estimation procedures for linear regression models:

$$\ln E = \ln a + b \ln L$$

When the quality focus measure cannot be considered interval (or even close to be interval) and is, for example, considered as an ordinal or a dichotomous response variable, then classification techniques such as logistic regression [HL89] can be used effectively.

When using a particular statistical technique, e.g., linear regression, to identify and formalize the relationship between a quality focus measure Y and an internal attribute measure X in order to validate X, one needs to show that X explains statistically a significant part of the variability of Y. However, X does not need to be a good predictor of Y since the objective here is only to provide statistical evidence of the existence of a dependency between the variables. For example, nobody expects to see coupling explain nearly all the variability of defect density because many other factors (e.g., human factors) are likely to have an impact. Therefore, it would not be reasonable to require that a coupling measure explain a very large percentage of the variability of defect density in order to be validated. Obtaining statistical significance should be the only objective to validate a measure.

This leads us to another issue: multivariate analysis. A particular internal attribute measure may not be significant as an isolated explanatory variable but may have a significant impact in a multivariate context, e.g., in an interaction term such as X_2X_3 in the multiple regression equation below:

$$Y = \alpha_0 + \alpha_1 X1 + \alpha_2 X_2 X_3$$

Where Y is the dependent variable, Xi's are the explanatory variables, and α_i's are the regression coefficients.

The variable X_3 appears in the interaction term but may not be significant by itself. This means, assuming that α_2 and X_2 are positive, that a larger X_3 (e.g., any product measure) will increase Y (e.g., defect density) for a given value of X_2. From a more general perspective, it would not be realistic to expect all product internal attribute measures being related to the quality focus measure to be significant explanatory variables in a univariate model.

The drawback is that multivariate analysis is more complex than univariate analysis. In a multivariate context, it often becomes very difficult to interpret regression parameters and models may show instability, e.g., due to outliers which are more difficult to detect in this context. In addition, interaction terms may be significant and will make the model even more difficult to interpret. This is in part to construct models that are easier to interpret and use that techniques such as classification trees [SP88] and Optimized Set Reduction [BBT93; J95] have been developed. Therefore, it is important to note that regression, although the most often used approach to mutlivariate software engineering model construction, is not the only alternative. Machine learning techniques, such as the ones mentioned above, should be seriously considered when performing multivariate analysis, especially in an exploratory context. For further discussion, refer to [BBT92].

4.4 Interpreting a Lack of Relationship

During empirical validation, if there is no empirical evidence supporting the expected relationship between the internal attribute and the external attribute (i.e., the relationship is not statistically significant), then the analyst may be faced with an interpretation difficulty. The difficulty is in deciding whether the internal attribute is valid. Below we present four issues that ought to be considered in such a situation.

4.4.1 Methodological Flaws

Two main factors, which we have termed here methodological flaws, may have contributed towards a lack of significance in the studied relationship(s). The analyst should rule these out before concluding that a measure is not valid.

The first factor is the design of the empirical study that generated the data for the validation. There may have been problems in the study itself which would explain the lack of relationship. For example, if an experimental design was employed, then the analyst should investigate the possibility that some confounding variables that have an impact on the results were not appropriately controlled. This would be a problem with the internal validity of the study. Also if, for instance, the study was conducted with university students and small programming tasks, then the analyst should consider whether the expected relationship(s) are most likely to exist only in an industrial environment with more experienced programmers and large scale programming tasks.

The second factor concerns the characteristics of the data that were collected. For example, if maintainability data on easily maintainable programs were collected, then it is likely that there would be little variation in the maintainability variable. If the analyst is using a Pearson product moment correlation to investigate the relationship, then such a restriction in range would lead to smaller relationships. Also, as another example, if there are extreme outliers in the data, depending on the method of analysis, these may have a substantial impact on the strength of the relationship.

4.4.2 **Small Sample Size**

Given that our criterion for the empirical validation of a measure is statistical significance, then the sample size can have a substantial impact on an analyst's findings and conclusions. This is because of statistical power. The power of a statistical test is defined as the probability of correctly rejecting the null hypothesis. The larger the sample size, the greater the statistical power of a given test. This means that if the analyst increases his/her sample size, and assuming the magnitude of the relationship remains unchanged, then s/he has a greater probability of finding the relationship statistically significant. This also means that if no relationship is identified, one possible reason is that the statistical test was not powerful (or sensitive) enough to detect it.

If an analyst does not find a statistically significant relationship, s/he should at a minimum determine whether the statistical test used was powerful enough for the given sample size. If s/he finds that the test was not powerful enough, then s/he should consider collecting more data and hence increasing the sample size. Often, however, that is not feasible. Alternatively, the analyst should consider using a more powerful test.

In general, parametric statistics are more powerful than their nonparametric counterparts. Figure 2 shows the sample sizes necessary for different magnitudes of two commonly used correlation coefficients, one parametric (Pearson's product moment correlation) and one nonparametric (Spearman's rank correlation)[4,5,6]. As can be seen, for a specified level of power, Spearman's correlation always requires a larger sample size than Pearson's coefficient (approximately 20% larger for strong relationships). In general, for *large samples*, to achieve the same power as the Spearman correlation, a test using Pearson's coefficient would require only approximately 91% of the former's sample size [SC88]. This is called the asymptotic relative efficiency (ARE) [Gib71].

---

[4] The values in this table are based on the tables provided in [KT87] and [Coh88].

[5] The calculations of sample sizes assume that the assumptions of the tests are met.

[6] Where there are analogous tables in [Coh88], the sample size values are only slightly different from [KT87] (approximately ±2 difference).

	Power = 90%			Power = 80%		
Corr.	**Pearson**	**Spearman**	**%Difference**	**Pearson**	**Spearman**	**% Difference**
0.1	854	1013	84%	618	733	84%
0.2	212	250	85%	154	183	84%
0.3	93	107	87%	68	79	86%
0.4	51	62	82%	38	46	83%
0.5	32	39	82%	24	30	80%
0.6	21	26	81%	16	20	80%
0.7	15	19	79%	12	15	80%

Figure 2: Minimal sample sizes required for Pearson's and Spearman's correlations for two levels of power (90% and 80%) at one tailed alpha = 0.05.

Therefore, in selecting a test to determine the statistical significance of a relationship, the analyst is more likely to be successful if s/he uses a parametric test. Of course, the assumptions of the particular test and the extent to which a particular set of data violate them should be considered when selecting a test.

4.4.3 Invalid Theory

The approach that we have presented in this paper for empirically validating measures of internal product attributes makes three assumptions:

1. that the internal attribute A1 is related to the external attribute A2
2. that measure X1 measures the internal attribute A1
3. that measure X2 measures the external attribute A2

Therefore, if the analyst finds a relationship between X1 and X2, then s/he has validated X1. The analyst can ensure that assumption 2 is met by theoretically validating X1 as described earlier in this paper. A similar procedure may be followed for ensuring that assumption 3 is met.

If the analyst does not find a relationship between X1 and X2, then s/he should consider questioning assumption 1. One possible reason for not finding a relationship between the measured variables is that the hypothesized relationship between the attributes A1 and A2 is incorrect.

4.4.4 Unreliable Measures

In some cases, measures of internal attributes are not fully automated and hence they involve a level of subjectivity. A good example of this is the Function Point measure of the functionality attribute. An important consideration for such measures is their reliability. Reliability is concerned with the extent to which a measure is repeatable and consistent. For example, whether two independent raters will produce the same Function Point count for the same system is a question of reliability.

Less than perfect reliability of measured variables reduces the magnitude of their relationship, and hence reduces the possibility of finding the relationship statistically significant. If an estimate of the reliability of a measure is available, then one can correct the magnitude of the relationship for attenuation due to unreliability (for example, see [Nun78][AW91]).

If the expected relationship involves measures that are not perfectly reliable, then the analyst should consider the possibility that attenuation due to unreliability is contributing towards the lack of significance. A correction for attenuation is most useful in terms of validation when the reliability of the measure(s) is quite low.

5. Conclusion

In this paper, we have presented an approach to validate measures of internal product attributes which integrates some existing approaches to validation, measurement theory, and the GQM paradigm. This is extremely important in the context of software engineering measurement where ad-hoc approaches to measurement are still common. Since building empirical relational systems is a crucial part of measurement, more empirical studies and replicated experiments are needed for our community to build an empirical understanding of the phenomena we are studying.

From the discussion above, it is clear that both empirical and theoretical validations, as they are defined above, are necessary and complementary. We think that even though a particular internal product attribute measure is only used to assess a product and not to predict any of its attributes, this particular measure is implicitly or explicitly associated to some measure of an external quality attribute. Therefore, such assumptions needs to be empirically validated.

Furthermore, the two validation types are related. In selecting an external attribute measure as a dependent variable to validate the usefulness of an internal product attribute measure, one is usually guided by an assumed relationship between the attributes. Consequently, if the data analysis results do not support the relationship, and after ruling out serious methodological problems, then one would not know whether it is the assumed relationship(s) or the measurement assumptions that are incorrect, or both. It would therefore be prudent to provide evidence that the internal product attribute measure is measuring what it is purporting to measure prior to validating its usefulness. One way of doing so is to capture the measurement assumptions by defining an empirical relational system as described in Section 3.1. Furthermore, let us assume that we define a design complexity measure X, a product quality measure Y, and that the data analysis results do support the relationship. In the case where it is not clear whether X measures design complexity, then the relationship may be spurious; and if it is not clear whether Y measures software quality, then the validation may be of no practical consequence.

It is also important to note that such validations are extremely difficult and one should not expect a single researcher to provide, within the context of one study, a complete and definitive validation. It is through replicated and iterative studies that confidence can be built over time.

6. References

[AW91] L. Aiken and S. West: *Multiple Regression: Testing and Interpreting Interactions*, Sage Publications, 1991.

[AKD+81] F. Andrews, L. Klem, T. Davidson, P. O'Malley, and W. Rodgers: *A Guide for Selecting Statistical Techniques for Analyzing Social Science Data*, Institute for Social Research, University of Michigan, 1981.

[BB81] J. Bailey and V. Basili: "A Meta-Model for Software Development Resource Expenditures." In *Proceedings of the International Conference on Software Engineering*, pages 107-116, 1981.

[Bas80] V. Basili: "Resource Models. " In *Tutorial on Models and Metrics for Software Management and Engineering*, IEEE Computer Society Press, V. Basili (ed.), 1980.

[B92] V. Basili: "Software Modeling and Measurement: The Goal/Question/Metric Paradigm." *Technical Report*, CS-TR-2956, University of Maryland, September 1992.

[BBT94] V. Basili, L. Briand, W. Thomas: "Experience Domain Analysis for Software Reuse", In *Proceedings of the 19th NASA-GSFC SEL Software Engineering Workshop*, 1994.

[BR88] V. Basili and D. Rombach: "The TAME Project: Towards Improvement Oriented Software Environments." In *IEEE Transactions on Software Engineering*, 14(6):758-773, June 1988.

[BBT92] L. Briand, V. Basili, W. Thomas: "A Pattern Recognition Approach for Software Engineering Data Analysis", In *IEEE Transactions on Software Engineering*, 18(11):931-942, 1992.

[BBT93] L. Briand, V. Basili and C. Hetmanski: "Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components," *IEEE Trans. Software Eng.*, 19 (11), November, 1993.

[BEM95] L. Briand, K. El Emam, and S. Morasca: "On the Application of Measurement Theory to Software Engineering." To appear in *Empirical Software Engineering Journal*, 1(1), 1995.

[BMB94(a)] L. Briand, S. Morasca, and V. Basili: "Defining and Validating High-Level Design Metrics." *Technical Report*, CS-TR-3301, University of Maryland, November 1994. Submitted for publication.

[BMB94(b)] L. Briand, S. Morasca, and V. Basili: “Property Based Software Engineering Measurement.” *Technical Report*, CS-TR-3368, University of Maryland, December 1994. Submitted for publication.

[Coh65] J. Cohen: “Some Statistical Issues in Psychological Research.” In *Handbook of Clinical Psychology*, B. Woleman (ed.), McGraw-Hill, 1965.

[Coh88] J. Cohen: *Statistical Power Analysis for the Behavioral Sciences*, Lawrence Erlbaum Associates, 1988.

[CC83] J. Cohen and P. Cohen: *Applied Multiple Regression / Correlation Analysis for the Behavioral Sciences*, Lawrence Erlbaum Associates, 1983.

[CG93] R. Courtney and D. Gustafson: “Shotgun Correlations in Software Measures”, *Software Engineering Journal*, 8(1):5-13, 1993.

[DG84] W. Dillon and M. Goldstein: "Multivariate Analysis: Methods and Applications." Wiley & Sons, 1984.

[F91] N. Fenton: *Software Metrics: A Rigorous Approach*, Chapman & Hall, 1991.

[FK90] N. Fenton and B. Kitchenham: “Validating Software Measures.” *In Journal of Software Testing, Verification and Reliability*, 1(2):27-42, 1990.

[Gib71] J. Gibbons: *Nonparametric Statistical Inference*, McGraw-Hill, 1971.

[HL89] D. Hosmer and S. Lemeshow: *“Applied Logistic Regression.”*, Wiley-Interscience, 1989

[J95] M. Jørgensen: "Experience with the Accuracy of Software Maintenance Task Effort Prediction Models," *IEEE Trans. Software Eng.*, 21 (8), August, 1995.

[KT87] H. Kraemer and S. Thiemann: *How Many Subjects? Statistical Power Analysis in Research*, Sage Publications, 1987.

[L91] K. B. Lakshmanan, S. Jayaprakash, and P. K. Sinha: "Properties of Control-Flow Complexity Measures," *In IEEE Transactions on Software Engineering*, vol. 17, no. 12, pp. 1289-1295, Dec. 1991.

[MT77] F. Mosteller and J. Tukey: *Data Analysis and Regression*, Addison-Wesley, 1977.

[Nun78] J. Nunnally: *Psychometric Theory*, McGraw-Hill, 1978.

[P72] D. Parnas: "On the Criteria to be Used in Decomposing Systems into Modules.", *Communications of the ACM*, 14(1): 221-227, 1972.

[R79] F. Roberts: *Measurement Theory with Applications to Decisionmaking, Utility, and the Social Sciences*, Addison-Wesley, 1979.

[S92] N. Schneidewind: “Methodology for Validating Software Metrics.” *In IEEE Transactions on Software Engineering*, 18(5):410-422, 1992.

[SP88] R. Selby and A. Porter: "Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis", *IEEE Trans. Software Eng.*, 14 (12), December, 1988.

[SC88] S. Siegel and J. Castellan: *Nonparametric Statistics for the Behavioral Sciences*, McGraw Hill, 1988.

[Ste46] S. Stevens: “On the Theory of Scales of Measurement.” In *Science*, 103(2684):677-680, June 1946.

[TZ92] J. Tian and M. V. Zelkowitz: "A Formal Program Complexity Model and Its Application," *J. Syst. Software*, vol. 17, pp. 253-266, 1992.

[Tuk86a] J. Tukey: “Data Analysis and Behavioral Science or Learning to Bear the Quantitative Man’s Burden by Shunning Badmandments.” In *The Collected Works of John W. Tukey*, Vol. III, Wadsworth, 1986.

[Tuk86b] J. Tukey: “The Future of Data Analysis.” In *The Collected Works of John W. Tukey*, Vol. III, Wadsworth, 1986.

[W88] E. Weyuker: “Evaluating Software Complexity Measures.” In *IEEE Transactions on Software Engineering*, 14(9):1357-1365, 1988.

[Z91] H. Zuse: *Software Complexity: Measures and Methods*, de Gruyter, 1991.