Metrics Validation Proposals: A Structured Analysis

Jean-Philippe Jacquet
E.mail: c3401@er.uqam.ca
Tel: (514) 987-3000 (6667)
Fax: (514) 987-8477

Alain Abran
E.mail: abran.alain@uqam.ca
Tel: (514) 987-3000 (8900)
Fax: (514) 987-8477

Research Lab. in Software Engineering Management,
Department of Computer Science,
Université du Québec à Montréal,
P.O. Box 8888, Succ. Centre-Ville,
Montréal (Québec), Canada H3C 3P8

Abstract:
In the literature, the expression *metrics validation* is used in many ways with different meanings. This paper analyzes and compares some of the validation approaches currently proposed. The basis for this analysis is a process model for software measurement methods which identifies the distinct steps involved from the design of a measurement method to the exploitation of the measurement results. This process model for software measurement methods is used to position various authors' validation criteria according to the measurement process to which they apply. This positioning enables the establishment of relationships among the various validation approaches. It also makes it possible to show that, because none of these validation approaches proposed to date in the literature covers the full spectrum of the process of measurement methods, a complete and practical validation framework does not yet exist.

Keywords : Validation, Software Measurement Methods, Software Metrics, Software Measure.

1. Introduction

Over the past twenty years, a significant number of software metrics have been proposed to better control and understand software development practices and products. Unfortunately, very few of them have been looked at closely from a measurement method perspective and it is currently difficult to analyze the quality of these metrics because of a lack of an agreed-upon validation framework.

What are software metrics and how do you determine that they are valid? A number of authors in software metrics have attempted to answer these questions [1, 2, 3, 4, 8, 9, 10, 11, 15, etc.]. However, the validation problem has up to now been tackled from different points of view (mathematical, empirical, etc.) and by giving different interpretations to the expression "metrics validation"; as suggested by Kitchenham et al: **"***What has been missing so far is a proper discussion of relationships among the different approaches*" [8].

This paper analyzes and compares the validation approaches currently proposed in the literature. The basis for this analysis is the process model for software measurement methods presented in [7]. This measurement method process model identifies the distinct steps involved, from the design of a measurement method to the exploitation of the measurement results in subsequent models, such as quality and estimation models.

Our proposed process model for software measurement methods is then used to position various authors' validation criteria according to these measurement process steps. This positioning enables the establishment of relationships among the various validation approaches. It also makes it possible to show that, because none of these validation approaches proposed to date in the literature covers the full spectrum of the process of measurement methods, a complete and practical validation framework does not yet exist.

This paper is organized in the following way. In the first section, the measurement process model is presented. The second section describes the different types of validation according to which part of the measurement process it refers to: the design of a measurement method, the application of a measurement method, or the exploitation of the measurement results (in predictive systems, for instance). In sections 4, 5 and 6, validation proposals from various authors are then positioned within the framework proposed.

2. A process model for software measurement methods

The analysis proposed here is being carried out by using a process model for software measurement methods. This process model [7] specifies the distinct steps involved from the design of a measurement method to its utilization. These steps are presented in Figure 1:

- **Step 1:** Design of the measurement method: before measuring, it is necessary to design a measurement method.
- **Step 2:** Application of the measurement method rules: the rules of the measurement method are applied to software or piece of software.
- **Step 3:** Measurement result analysis: the application of the measurement method rules produces a result.
- **Step 4:** Exploitation of the measurement result: the measurement result is exploited in a quantitative or qualitative model.

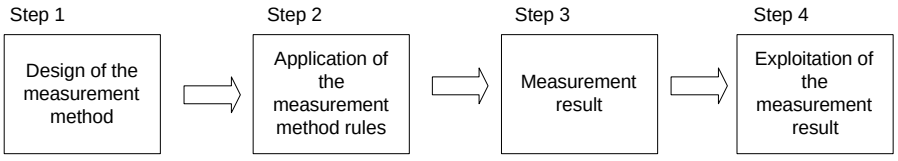

Figure 1: Measurement Process - High-level Model (Jacquet and Abran 1997 [7])

This high-level model was then refined based on a literature review in the domain of scientific measurement. This literature review, from within and from outside the software engineering domain, has permitted the identification of the required substeps within each of the proposed measurement steps.

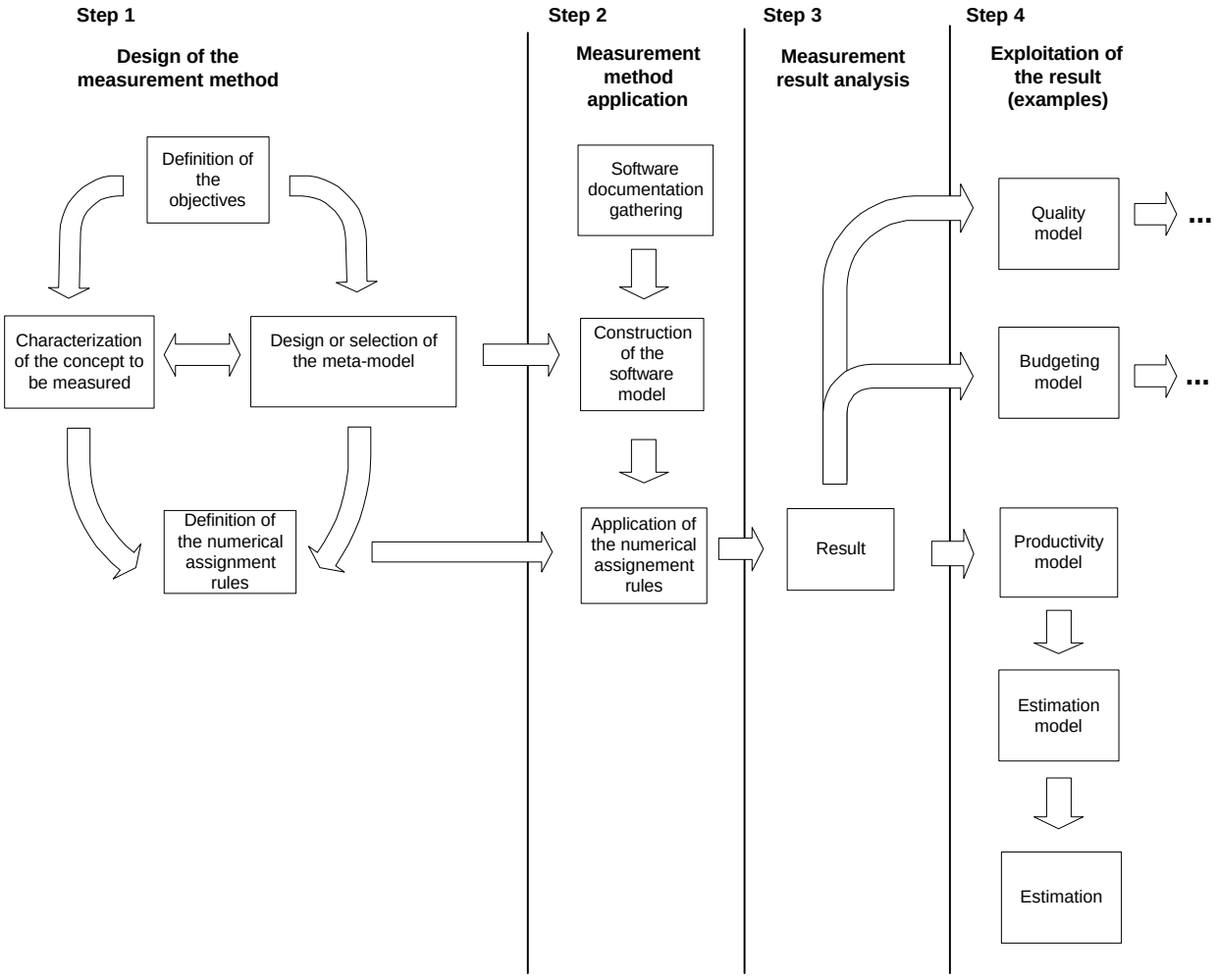

Figure 2: Measurement Process - Detailed Model (Jacquet and Abran 1997 [7])

The detailed set of substeps identified is illustrated in Figure 2 and these substeps can be described as follows:

Step 1: Design of the measurement method

- Substep 1 : For the initial substep, the objectives of the measurement method must be specified. Among other things, what is going to be measured (what object, what attribute, etc.) must be known precisely, what will the intended uses of the measurement method be, etc.

- Substep 2: Once the attribute (or concept) has been chosen, an (empirical) operational definition of this attribute must be given. This can easily be done for

concrete attributes (such as size for a person, or size in lines of codes for software), but will be more complicated for abstract attributes (this is what Zuse calls the "*intelligence barrier*" [16]). In that case, the operational definition has to be as close as possible to the "meaning" of the attribute in order to capture it properly. Using mathematical notation (measurement theory vocabulary), the axiomization of the attribute will constitute an empirical relational set.

- Substep 3 : Design or selection of the metamodel
Software is not a tangible product. However, it can be made visible through multiple representations (e.g. for a user, a set of reports, screens etc; for a programmer, a set of lines of Code, etc.). The set of characteristics selected to represent software or a piece of software, and the set of their relationships, constitute the metamodel proposed for the description of the software to which the proposed measurement method will be applied. The metamodel must therefore describe the entity types that will be used to describe the software and the rules that allow the identification of the entity types. For example, in the Function Point measurement method (FPMM) [17], an Internal Logical File (ILF) is a piece of software of the entity type. This entity type is defined according to the FPMM's metamodel. The IFPUG manual presents a definition of this ILF entity type, as well as identifying rules to ensure that each and every ILF can be clearly recognized within software.

- Substep 4 : Definition of the numerical assignment rules
In this substep, the rules allowing assignment of a numerical value to the couple (attribute, object) measured are defined. Very often, the basis for the numerical assignment rules is the characterization of the concept and the proposed metamodel. In mathematical vocabulary, the expression of these rules allows definition of a formal relational system.

Step 2: Application of the measurement method
This step is made up of three substeps:

- Substep 1: the gathering of the documentation necessary to carry out the application of the measurement method.
- Substep 2: the construction of the software model (for example, the various pertinent entities for the measurement method are referenced, on the basis of the meta-model defined or selected in the design step).
- Substep 3: the application of the numerical assignement rules.

Step 3: The application of the measurement method provides a measurement result.

Step 4: This result of the measurement method can be used in predictive models such as budgeting or quality models.

3. What is metrics validation?

The expression *metrics validation* has been used with many distinct interpretations in the software engineering literature, leaving readers somewhat puzzled as to which authors' proposals should be used, and under what circumstances. Rather than trying to recommend and select a single author's interpretation as the correct one for what has been referred to by various authors as "metrics validation", we will look at the validation issue using the measurement process model presented in the previous section. Therefore, rather than discussing a single validation proposal, a framework will be suggested which includes a sequence of validation procedures which should address, with different techniques, the distinct steps of the measurement process model, as presented in Figure 2:

• **Validation of the design of a measurement method.**
This type of validation deals mainly with step 1 of our model. It consists in checking that the measurement method really measures what it is supposed to measure.

For example, Fenton [3] refers to this concept when he discusses the validation of "a software measure ", expressing it in the following way: *Validation of a software measure is the process of ensuring that the measure is a proper numerical characterization of the claimed attribute[...].*

Similarly, Zuse [16] refers to this validation concept of the design of a measurement method when he uses the expression " internal validation of a software measure": *Internal validation of a software measure means to check empirically whether the measure really captures the claimed attribute.*

• **Validation of the application of a measurement method**. This type of validation addresses both step 2 and step 3 of the measurement process model presented in Figure 2.

This type of validation aims to answer questions such as: How can one know whether the application of a specific measurement has been properly carried out? and, "What degree of confidence one can have in the result obtained by applying a specific measurement method?.

For this kind of operation, the use of the word "validation" is not unanimously accepted. Now, such validation is crucial, but, surprisingly, is scarcely discussed in the software engineering literature.

• **Validation of predictive systems.** This type of validation deals with the predictive quality of the models in which the measurement results are exploited, as illustrated in the right-hand side of Figure 2.

This is the type of validation most often discussed in software engineering. In this case, the expression *software metrics* is taken to meant a *predictive system* or *integrated in a predictive system,* but does not deal at all with the issues related to the design and application of measurement methods.

Fenton [3], for example, addresses the validation of predictive system: *Validation of a prediction system, in a given environment, is the process of establishing the accuracy of the prediction system by empirical means, i.e. by comparing model performance with known data points in the given environment.*

In the software engineering literature, the distinction between the above three types of validation is almost never explicitly stated. For example, fairly distinct validation methods are proposed for "validating metrics", but even though they explicitly refer to this same expression they do not address the same issue (validation of measurement methods and validation of predictive systems) and they do not use the same validation techniques.

In the next section we will present and discuss some work that has been carried out on each of these kinds of validation and will try to highlight and clarify the differences *between validation of a measurement method* and *validation of a predictive system*.

4. Validation of the design of a measurement method

What is a valid measurement method? This question has not been addressed directly by most authors. Some authors have, however addressed it indirectly (for example: Fenton [3], Zuse [16]) when they argue that a software measure is valid if the representation condition is verified, i.e. if a homomorphism exists between the empirical relational system (defined at the second substep of step 1 in the process of Figure 2) and the numerical relational set (defined at the fourth substep in the process in Figure 3). When they do so, they are addressing the design step of a measurement method.

The validation of the design of a measurement method should therefore deal with the establishment of relations across the following substeps: characterization of the concept to be measured and definition of the numerical assignment rules. A valid measurement method design would consist of a design for which its numerical assignment rules properly represent the empirical characterization of the measured attribute, and the validation process would consist in proving that empirical data are properly "captured" by the measurement method design.

Various authors have proposed more detailed validation criteria which address this measurement method design issue. These criteria are mostly based on checking that the representation condition is satisfied, but also that this is not a sufficient condition. For example, in Shepperd et al.'s work [12, p.75], inspired by Weyuker [13] and Prather[10], a list of axioms is proposed which must be satisfied. Some of these axioms are:

Axiom a: *It must be possible to describe, even if not formally, the rules governing the measurement method [...].*
Axiom b: *The measure*[1] *generate at least two equivalence classes.*
Axiom c: *If an infinite number of objects or events is measured, eventually two or more must be assigned to the same equivalence class, that is, you can't measure an infinite number of objects.*
Axiom d: *The metric*[1] *must not produce anomalies, that is, the metric must preserve empirical ordering. In other words, the representation theorem must hold.*

The fifth axiom deals with the representation condition but the others can be seen as complementary axioms. For example, axiom 2 is aimed at checking that the measurement method "does something", i.e. is capable of discrimination and will not assign the same value to every object. Shepperd et al. argue then that axiomization is a *vital tool* for validation because it provides, among other things, ways to establish some properties of the model, such as consistency - *so that there exists one and only one outcome for any set of inputs* - and completeness - *the axiom set is sufficiently rich and that there is no set of inputs for which no outcome is prescribed*.

This type of validation requirements reflects the fact that, even if the representation condition is satisfied, a measurement method could have undesirable properties. For example, many of these properties can manifest themselves because of the construction of the (empirical) operational definition of the concept. For example, supposing that the representation theorem has been demonstrated for a functional size measurement method, then this method will assign the same number to every software if and only if the empirical relational system asserts that every software has the same functional size. This means that this empirical relational system, and consequently the operational definition of the concept, does not characterize the attribute of functionality properly (substep 2 of step 1 in Figure 2). In that case, it cannot be said that the measurement method measures what it is supposed to measure. Consequently, validation of a measurement method also has to deal with validation of the other substeps of the design step (in Figure 2), including the characterization of the attribute to be measured.

Kitchenham et al. have proposed additional criteria in [8] which deal with the design step of a measurement method. In what they refer to as validation of a measure**,** they describe and discuss several theoretical and empirical validation criteria for the design of a measurement method. Among other things they propose to:

- check attribute validity (substeps "definition of the objectives" and "characterization of the attribute to be measured" in Figure 2). This means, for example, checking whether or not the attribute is exhibited by the measured object. They also propose empirical methods to check whether or not the results of a measurement method capture the meaning of the attribute. Their proposed methods use measures of association and are validation methods

[1] In this reference, the words *measure* and *metrics* can be understood as *measurement methods*.

based on the establishment of correlations between the measured attribute and results of the measurement method.

- tackle validation of the "numerical assignment". This mean requiring mathematical consistency of these rules (for both direct and indirect measures). They also propose models and methods to validate units of measure.

The set of validation criteria proposed by Kitchenham et al. addresses many of the substeps of step 1 (Figure 2). Nevertheless, there is at least one that is not well covered, one that should deal with objects (software) or, more precisely with the representation of these objects (substep 3 of step 1 in Figure 2) on which the measurement is applied. For example, for the application of the Function Point Analysis measurement method [17], the software object has to be modeled using concepts of ILF, EIF, boundary, etc.. Such a model is created using identification rules, and from that point on it allows the application of the numerical assignment rules. In other words, the identification rules have an influence on the creation of the numerical relational set and, logically, on the design of the empirical relational set as well when these rules are used in the design of the operational definition of the concept. For example, many " software metrics " definitions use terms like "process", "flows of data" without providing precise definitions for them. Perhaps the representation theorem has been demonstrated for these " metrics ", but, if the entities used in this demonstration are not properly and unambiguously defined, then what value can be credited to this demonstration?

In conclusion, one can say that in the software engineering literature, validation of the design of a measurement method has mainly been tackled from a measurement theory viewpoint: a valid measurement method is a method which verifies the representation theorem. Nevertheless, it appears that this requirement is not sufficient and some authors have proposed additional criteria. However, it seems that the validation of a measurement method has to do with the validation of an empirical relational set, and, consequently, with the validation of attributes and objects (or models of theses objects[2]) taking part in the description of this empirical set and consequently in the measurement method. This problem does not seem to have been tackled completely and the validation criteria for this substep of the design of a measurement method are still relatively poorly covered.

5. Validation of the application of a measurement method

The second type of validation is the validation of the application of a measurement method: it deals with a specific utilization of a measurement method. Once a measurement method has been validated, it can be applied. But, after the method has been applied, how can it be ascertained that the application process has been properly carried out? What degree of confidence can one have in the measurement result, knowing that

[2] In this, one can find the type of argument made by Gustafson et al. in [4] about the importance of modeling the object to be measured.

mistakes may have been made when applying the method? How can a result be formally accepted before it is used by an organization?

Even though these questions are important, they are rarely discussed in the software engineering literature.

Validation of the application of a measurement method should involve both steps 2 and 3 of the measurement process described in Figure 2:

- Validation of the quality of the implementation of the various substeps (software documentation gathering, construction of the software model, application of the numerical assignment rules) of the process of applying a measurement method.

- Validation of the correctness of the result obtained (step 3 in the process in Figure 2).

Among the very few proposals identified in the software engineering literature for the evaluation of the correctness of a measurement result, Morris and Desharnais [9] address this type of validation when they discuss the validation of Function Point measurement results. This proposal consists of two steps:

- *A priori validation : reviews all the steps and procedures of the data collection process within benchmarking data [9].* For example, a priori validation includes verification of accurate and reliable documentation and the *verification of the existence of training and/or coaching programs to ensure that the staff collecting the metrics have a solid understanding of software metrics that they collect.*

- *A posteriori validation: establishes the degree of quality of the data collected and, wherever required, will eliminate from the comparative studies the data deemed not reliable [9].* For example, a posteriori validation includes verification *that the data definitions and coding schemes are consistent across sites and over time.*

Morris and Desharnais validation proposal is fairly extensive for this type of validation but, it has been discussed only in the context of its application to the Function Point Analysis measurement method. Nevertheless, many of the criteria described in this validation method could be generalizable to other measurement methods.

6. Validation of a predictive system

The third type of validation is the validation of a predictive system. This type of validation is, in fact, the one most often discussed in software engineering and many papers have been written about validating metrics from the perspective of their use as predictive systems. A possible explanation could be that it is perceived to be more useful and easier to validate predictive systems because the properties of many external variables, such as cost, time and productivity, are better known.

Schneidewind writes that *if metrics are to be of greatest utility, the validation should be performed in terms of the quality function (quality assessment, control and prediction) that the metrics are to support* [11]. However, Kitchenham et al. disagree with Schneidewind and remark that *validating a predictive or usage model is different from validating a measure* [8]. It is then important to make a real distinction between validation of a measurement method and validation of a predictive system. For example, a predictive model using one or multiple measurement methods can be valid even thought one or more of the measurement methods are invalid. This would mean that the measurement methods and the model are self correctors (i.e. the errors of one counterbalance the errors of the other).

Furthermore, validation of a measurement method is less context-dependent than the validation of a predictive system. For example, one can validate a functional size measurement method for a type of software. Now, this measurement method can be used in a predictive system in order to predict productivity for a specific group of programmers in a specific environment. This predictive system can be validated for this context, but would have to be revalidated if used in another environment. This revalidation involves reexamination of the results of the predictive model in this new context and not the revalidation of the measurement method itself. Consequently predictive systems are validated according to a context. This type of validation should be performed every time the predictive system is used in a different context.

A great number of methods have been developed for the validation of predictive systems. Two representative examples are now briefly discussed.

Among the set of articles written about the validation of predictive systems, one of the most exhaustive is [11]. In this article, Schneidewind proposes an empirical validation method based on corroboration of the relationships among attributes in specific contexts.

Schneidewind's validation proposal for predictive systems is based on six validity criteria which are: association, consistency, discriminative power, tracking, predictability and repeatability (see [11]). For example, the repeatability criterion "*assesses whether* (a metric) *M can be validated on a sufficient percentage of trials to have confidence that it would be a dependable indicator of quality in the long run*". These six criteria support the functions of assessing, control and predicting. This proposal for this type of validation is mostly based on statistical techniques such as correlation analysis, providing specific methods to establish correlations among attributes.

Schneidewind's approach is mainly empirical, but does not refer at all to measurement theory and does not deal with measurement method concepts. On the other hand, in [16], Zuse has begun to addresses the validation of predictive systems from a measurement theory viewpoint. He proposes processes and theoretical conditions for validation of measurement methods as predictive systems.

In conclusion, one can say that the validation of predictive systems is concerned with the fourth column of the process presented in Figure 2. This kind of validation is very often confounded with the validation of measurement methods (first column of the process in Figure 2). This can be explained by the fact that many authors have not made a clear distinction between a measurement method and a predictive system.

7. Conclusion

In the software engineering literature, the expression *software metrics validation* has been used with different interpretations and most authors have not explicitly stated which step of a measurement method process their validation proposal is intended to address. Some classifications of validation studies for software metrics have been proposed in the past (see, for example, Gustafson [6]) but these classifications has been made according to different criteria and sometimes without a clear distinction being made between measurement methods and predictive models.

In this paper, a framework has been explicitly stated which characterizes three types of validation when addressing the validation issue: validation of the design of a measurement method, validation of the application of a measurement method and validation of the use of measurement results in a predictive system (see Figure 3).

VALIDATION TYPE	DESCRIPTION
Validation of the Design of a Measurement Method	This type of validation refers mainly to **step 1** of our process. In the software engineering literature, validation of the design of a measurement method has been tackled mostly from a measurement theory viewpoint: a valid measurement method is a method which verifies the representation theorem. However the validation of all the substeps of step 1 have not been addressed.
Validation of the Application of a Measurement Method.	The validation of the application of a measurement method has been rarely discussed in software engineering although this is an important issue for practitioners and for good empirical research. Morris et al. distinguishes *a priori* validation, which is related to **step 2** of the measurement process in Figure 2, from *a posteriori* validation, which is related to the **step 3** of the measurement process.
Validation of Predictive Systems	This type of validation refers to **step 4** of the process presented in Figure 2. This type of validation is very often confounded with validation of measurement methods (first column, Figure 2).

Fig 3: Validation Types

In this article, many authors have been referred to according to the validation type they refer to. A recapitulation of this classification is presented in Figure 4. It has also been illustrated that a great number of proposals to validate measurement methods are incomplete since they do not tackle the correctness of the operational definition of the attribute (and consequently the pertinence of the empirical set used) or the correctness of the object representation used by the measurement method. This problem could perhaps be addressed by investigating validation frameworks from other research fields such as the social sciences or management sciences for example.

MEASUREMENT PROCESS MODEL	AUTHORS WITH VALIDATION PROPOSALS
Step 1: Design of the Measurement Method.	
Definition of the objectives.	[8] Kitchenham et al.
Design or Selection of the Metamodel.	
Characterization of the Concept.	[8] Kictchenham et al.: for example, these authors tackle the attribute validity. [12] Shepperd, Ince, [10] Prather, [13] Weyuker: These authors propose axioms that should be satisfied. These axioms are in the main related to properties of the attribute (or concept) measured.
Definition of the Numerical Assignment Rules.	[3] Fenton, [15, 16] Zuse, [8] Kictchenham et al. [12] M. Shepperd, Darrel Ince, etc: Theses authors require that the representation condition be satisfied, i.e. that the numerical assignments rules properly characterize the attribute (concept) measured. [8] Kictchenham et al: for example, these authors tackle the validity of the measurement method unit.
Step 2: Measurement Method Application	
Software Documentation Gathering.	[9] Morris, Desharnais
Construction of the Software Model.	[9] Morris, Desharnais
Application of the Numerical Assignment Rules	[9] Morris, Desharnais

Step 3 : Measurement Result Analysis	[9] Morris, Desharnais
Step 4: Exploitation of the Results	[3] Fenton [8] Kitchenham et al. [11] Schneidewind [15] Zuse [16] Zuse

Fig. 4 Validation notions and authors

Acknowledgments

This research was carried out at the Software Engineering Management Research Lab. at the Université du Quebec à Montréal. This laboratory is made possible through a partnership with Bell Canada. Additional funding is provided by the Natural Sciences and Engineering Research Council of Canada. The opinions expressed in this article are solely those of the authors.

References

[1] A. Abran, E. Ahki.
Validation requirement for functional size measurement. Internal Report, Research Laboratory in Software Engineering, Université du Québec à Montréal, 1994.

[2] K.G. van der Berg, P.M. van der Broek
"Axiomatic Validation in the Software Metric Development Process", in Software Measurement, Edited by A. Melton, International Thomson Publishing Compagny, p.157, 1996.

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

[4] N. Fenton and B. Kitchenham, "Validating Software Measures", J. of Software Technology, Verification and Reliability, vol 1, no. 2, pp. 27-42, 1991.

[5] D.A. Gustafson, J.T. Tan, P. Weaver
"Software Metric Specification", in Software Measurement, Edited by A. Melton, International Thomson Publishing Compagny, p.179, 1996.

[6] D.A. Gustafson, R.M. Toledo, N. Temsamani
"A critique of validation/verification techniques for software development measures", in T. Denvir, R. Herman and R. W. Whitty, eds., Formal Aspects of Measurement, pp. 145-156. Springer, New York, 1992.

[7] J.P. Jacquet, A. Abran
"From Software Metrics to Software Measurement Methods: A Process Model". Accepted at the Third International Symposium and Forum on Software Engineering Standards, ISESS '97, IEEE, Walnut Creek (CA), June 1997.

[8] B. Kitchenham, S. L. Pfleeger, N. Fenton.
"Towards a Framework for Software Measurement Validation", IEEE Transactions On Software Engineering, Vol 21, Dec. 1995.

[9] P.M. Morris, J.M. Desharnais
"Function Point Analysis. Validating the Results", IFPUG Spring Conference, Atlanta, April 1996.

[10] R.E. Prather.
"An Axiomatic Theory of Software Complexity Metrics". Computer Journal, 27(4):42-45, 1984.

[11] N. Schneidewind
"Methodology for Validating Software Metrics", IEEE Transactions on Software Engineering, Vol. 18, no. 5, pp. 410-442, May 92.

[12] M. Shepperd, Darrel Ince
"Derivation and Validation of Software Metrics", Oxford Science Publications, 1993.

[13] E.J. Weyuker
"Evaluating Software Complexity Measures", IEEE Transaction of Software Engineering, 14(9) :1357-1365, 1988.

[14] Horst Zuse
Foundations of Validation, Prediction and Software Measures, Proceedings of the AOSW94 (Annual Oregon Software Metric Workshop), Portland, April 20-22, 1994.

[15] Horst Zuse
"Measurement theory and software measures", Formal Aspects of Measurement, Editors: T. Denvir, R. Herman, R. Whitty. Workshops in Computing, Springer Publisher, 1992.

[16] Horst Zuse
"A Framework of Software Measurement", preliminary version, Oct. 96, to be published in 1997.

[17] International Function Point Users Group (IFPUG)
"Function Point Counting Practices Manual", Version 4.0, 1994.