Validating Software Metrics: A Spectrum of Philosophies

ANDREW MENEELY, BEN SMITH, and LAURIE WILLIAMS, North Carolina State University

Context. Researchers proposing a new metric have the burden of proof to demonstrate to the research community that the metric is acceptable in its intended use. This burden of proof is provided through the multi-faceted, scientific, and objective process of software metrics validation. Over the last 40 years, however, researchers have debated what constitutes a "valid" metric.

Aim. The debate over what constitutes a valid metric centers on software metrics validation criteria. The objective of this article is to guide researchers in making sound contributions to the field of software engineering metrics by providing a practical summary of the metrics validation criteria found in the academic literature.

Method. We conducted a systematic literature review that began with 2,288 papers and ultimately focused on 20 papers. After extracting 47 unique validation criteria from these 20 papers, we performed a comparative analysis to explore the relationships amongst the criteria.

Results. Our 47 validation criteria represent a diverse view of what constitutes a valid metric. We present an analysis of the criteria's categorization, conflicts, common themes, and philosophical motivations behind the validation criteria.

Conclusions. Although the 47 validation criteria are not conflict-free, the diversity of motivations and philosophies behind the validation criteria indicates that metrics validation is complex. Researchers proposing new metrics should consider the applicability of the validation criteria in terms of our categorization and analysis. Rather than arbitrarily choosing validation criteria for each metric, researchers should choose criteria that can confirm that the metric is appropriate for its intended use. We conclude that metrics validation criteria provide answers to questions that researchers have about the merits and limitations of a metric.

Categories and Subject Descriptors: D.2.8 [**Software Engineering**]: Metrics—*Product metrics, Process metrics*

General Terms: Measurement, Theory

Additional Key Words and Phrases: Software metrics, validation criterion, systematic literature review

ACM Reference Format:
Meneely, A., Smith, B., and Williams, L. 2012. Validating software metrics: A spectrum of philosophies. ACM Trans. Softw. Eng. Methodol. 21, 4, Article 24 (November 2012), 28 pages.
DOI = 10.1145/2377656.2377661 http://doi.acm.org/10.1145/2377656.2377661

1. INTRODUCTION

Practitioners and researchers alike use software metrics to both improve and understand software products and the software development processes.

A. Meneely is currently affiliated with Rochester Institute of Technology. B. Smith is currently affiliated with IBM.
Authors' addresses: A. Meneely, Department of Software Engineering, Rochester Institute of Technology; email: andy@se.rit.edu; B. Smith, IBM Software Group; email: benjsmi@us.ibm.com; L. Williams, Department of Computer Science, North Carolina State University, Raleigh, NC; email: laurie_williams@ncsu.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2012 ACM 1049-331X/2012/11-ART24 $15.00
DOI 10.1145/2377656.2377661 http://doi.acm.org/10.1145/2377656.2377661

The field of software metrics has a variety of applications including quality assessment, prediction, task planning, and research. Researchers proposing a new metric have the burden of proof to demonstrate to the research community that the metric is truly representative of the attribute it is intended to represent.

But how do we, as a community, ensure that a metric is suitable and acceptable for its intended purpose? Without some formal system of rules for the determining the merits of a metric, the software engineering community could find itself flooded with metrics that lend no understanding to the state of the art. This system of rules for ensuring the worthiness of a metric is known as *software metrics validation*, and software engineering researchers have debated what constitutes validation for almost half a century.

The community has not yet reached a consensus on this system of rules. Instead, software metrics researchers have often been proposing their own, specialized means of validation. This ad hoc approach to validation leads to results that cannot be generalized and to contributions that are not stated in a manner consistent with a standard form of metric validation.

Before we look at where metric validation must go, we must look at where it has been. The metrics validation community has likely not reached a consensus because no researcher has provided a "proper discussion of relationships among the different approaches [to metric validation]" [Kitchenham et al. 1995]. The objective of this article is to guide researchers in making sound contributions to the field of software engineering metrics by providing a practical summary of the metrics validation criteria found in the academic literature. We performed a systematic literature review, beginning with 2,288 potential peer-reviewed publications and ultimately focused on 20 publications that propose or indicate software metrics validation criteria. Our review is a useful guide for two audiences: (1) metrics researchers or software engineering researchers who propose, use, and validate new metrics; and (2) metric validation researchers or software engineering researchers seeking to propose methodologies, criteria, or frameworks for validating metrics.

Metric researchers will want to use this review for the follwing reasons.

—*A reference guide*. We have compiled a list of the unique criteria presented in the literature with definitions and examples. Additionally, metrics researchers can see where authors discuss the same criteria but with different vocabulary.
—*A source for threats to metric validity*. Metrics researchers can consult our survey to enumerate the issues a given metric may encounter. Furthermore, this source acts as a guide on issues where the community disagrees or published criteria contradict each other.
—*A big picture*. We present a hierarchical relationship among the criteria that can be used to gain an understanding of how a given criterion relates to the "big picture" of software metric validation. Our analysis of the philosophical motivations behind a validation criterion is helpful for understanding not only the criteria themselves, but the "spirit" of the criteria.
—*Inspiration*. The criteria that have been proposed can act as a set of guidelines to help inspire a new way of thinking about currently-existing metrics. Our comparative analysis of the criteria introduces underlying, tacit concepts that all researchers should be aware of when considering new metrics.

Software researchers will want to use this review for the follwing reasons.

—*A roadmap for current criteria*. We provide the validation criteria that researchers can use to view how their proposals fit into the overall body of work on metric validation.

—*A call for discussion*. In analyzing the validation criteria in the literature, we have summarized the discussion surrounding metric validation. We have found that this discussion has not concluded and wish to reinvigorate the discussion in the research community to help reach a consensus on what is meant by metric validity.
—*A direction for a standard*. Understanding the categorization and the motivation of the validation criteria assists the metrics validation community in deciding on a standard set of criteria for validating a metric.
—*A guide for future criteria*. The hierarchical relationship we discovered when comparing the criteria can serve as a tool for generating new metrics validation criteria. Areas of metric validation are missing in the overall hierarchy, and new criteria could be proposed that address these areas.

This article is organized as follows. In Section 2, we present the terms specific to this review and their definitions. Next, in Section 3 we present the process we used to conduct the review. We present the extracted criteria in Section 4 and the mapping of their relationships in Section 5. Section 6 outlines the philosophies of the two opposing motivations behind the criteria. In Section 7, we describe how to use a metric's intended use to choose the appropriate validation criteria. Finally, Section 8 concludes.

2. TERMS AND DEFINITIONS

During our review, we found many different usages and definitions for the same words, so we define our usage of these words in Section 2.1. Next, we define examples of metrics that we refer to throughout the article in Section 2.2.

2.1. Metric Terminology

—*Attribute*. The specific characteristic of the entity being measured (IEEE 1990). For example, the attribute for a Lines of Code metric is "size".
  —*Internal attribute*. Attributes that the product itself exhibits. For example, the size of code (ISO/IEC 1991) is an internal attribute [Fenton and Kitchenham 1991].
  —*External attribute*. Attributes that are dependent on the behavior of the system. For example, the reliability of a system is an external attribute [ISO/IEC 1991].
—*Component*. One of the parts that make up a software system. A component may be hardware or software and may be subdivided into other components [IEEE 1990].
—*Failure*. An event in which a system or system component does not perform a required function within specified limits [ISO/IEC 1991].
—*Fault*. An incorrect step, process, or data definition in a computer program [ISO/IEC 1991].
—*Metric*. A metric is a "quantitative scale and method which can be used to determine the value a feature takes for a specific software product" (IEEE 1990). Fenton [Fenton and Neil 2000; Fenton 1994] explains that essentially any software metric is an attempt to measure or predict some attribute (internal or external) of some product, process, or resource. Fenton and Kitchenham [1991] point out that metric has been used as:
  (i) a number derived from a product, process or resource;
  (ii) a scale of measurement
  (iii) an identifiable *attribute* (as defind).
  Also, a metric measures the degree to which a system, component or process possesses a given attribute (as defind) (IEEE 1990). An *internal metric* measures an internal attribute, and an *external metric* measures an external attribute.
—*Quality Factor*. An attribute of software that contributes to the degree to which software possesses a desired combination of characteristics [Schneidewind 1992].

—*Statistical correlation*. We use the term “statistical correlation” in the broad sense of one variable “corelating” with another, not to be confused with the correlation coefficients that are specific statistical tests used to estimate the correlation between two variables.
—*Validation*. The ISO/IEC of definition of validation is “confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled” [ISO/IEC 1991].

2.2. Examples

In our discussions throughout this article, we use five examples of metrics to illustrate concepts found in the criteria. We list the referenced definitions for these metrics here.

—*Lines of Code (LOC) metric:* The general notion of counting the number of lines of source code in a program, without adhering to one specific standard.
—*Cyclomatic Number*: McCabe’s cyclomatic complexity [McCabe 1976], which is based on the number of edges, nodes, and components in a program flow graph.
—*Code Churn*: A measure of how much a unit of code has changed over a period of time, usually measured by the number of lines of code added/deleted at each revision in the version control system [Elbaum and Munson 1998].
—*Static Analyzer*: A tool that performs automated static analysis on source or binary code, such as FindBugs [Ayewah et al. 2008]. An automated static analysis tool is a program that analyzes another program’s source code and reveals possible faults in the form of “warnings”.
—*Code Coverage*: The percentage of a structural entity that is executed by a given test set. For example statement coverage is the percentage of statements executed by the test suite.

3. METHODOLOGY AND PROCESS

Kitchenham [2004] recommends a procedure for conducting systematic literature reviews which we follow for this review. One goal of a systematic literature review is “to provide a framework/background in order to appropriately position new research activities” [Kitchenham 2004]. The following sections focus on the bottom two tiers: gathering the literature and analyzing the literature. The process is broken down into two parts: planning the review (Section 3.1) and conducting the review (Section 3.2).

3.1. Planning the Review

To begin our systematic literature review, we conducted a preliminary mapping study to identify our research objective, based on Kitchenham’s systematic review process [Kitchenham 2004]. As a result of this mapping study, the objective of this article is to guide researchers in making sound contributions to the field of software engineering metrics by providing a practical summary of the metrics validation criteria found in the academic literature.

Kitchenham [2010] also conducted a preliminary mapping study in the field of software metrics. The Kitchenmham study focuses on influential metrics papers, while our study focuses on metric validation. Specifically, the Kitchenham study identifies the most influential papers in the software metrics literature by analyzing their citation counts. We focused on extracting metrics validation criteria from the literature to summarize and analyze the existing literature.

The second step for planning a systematic literature review is to develop the search strategy, as a part of developing the review protocol [Kitchenham 2004]. As recommended in Brereton et al. [2007], we used the following search engines to conduct our review: Google Scholar (http://scholar.google.com); IEEExplore (http://ieeexplore.

ieee.org); CiteSeerX (http://citeseerx.ist.psu.edu/); ACM Portal (http://portal.acm.org/portal.cfm).

After narrowing our research objective to focus on the criteria for validating metrics, we decided on the following queries:

—software AND engineering AND metrics,
—software AND metrics AND validation,
—software AND metrics AND evaluation.

These search criteria capture a wide range of publications, so an overwhelming number of results were obtained with each search. Since reviewing all of the results was infeasible, we developed a stopping heuristic. We used our searches using the following method.

(1) For both of the search strings, run the search on all four search engines.
(2) Record the bibliographic information for any source not previously recorded that appeared to be about the validation (or evaluation) of software engineering metrics. This step was mostly based on title and type of publication.
(3) Continue through the search results until reaching 10 consecutive, irrelevant results. Since the search engines we used rank the results by relevance, we found this procedure to be effective at producing useful results. In most cases, the density of relevant titles steadily decreased as we examined each source.
(4) Otherwise, continue to the next set of 10 results (step 3).

We created the form shown in Figure 1 to assess the quality of our primary sources and identify sources that fit our research objective.

3.2. Conducting the Initial Search

The initial search consisted of a "first pass" (Section 3.2.1) through the titles, then a "second pass" (Section 3.2.2) through the abstracts of the given titles. The "third pass" was through the full texts of the chosen sources (Section 3.2.3).

3.2.1. First Pass: Authors and Titles. In the first pass through the search results, we were as inclusive as possible because we did not read abstracts; we tracked only titles and authors. The procedure resulted in the data presented in Table I. The total results column in Table I comes from the search engine's self-reported total number of results upon executing the query. The reviewed results column comes from executing the procedure in Section 3.1. The researchers are given code names: here, the first author is Researcher A and the second author is Researcher B. For example, Researcher B searched IEEExplore for "software AND metrics AND evaluation" and found a total of 1,836,783 results. Researcher B iterated through each set of 10 results and collected the relevant titles and authors for these sources. Then, after 90 sources, the researcher saw a set of 10 results (numbers 90–100) with no relevant titles, and the researcher stopped.

Out of the 3 million sources returned by the search engines, we examined the titles of 2,288 sources. We accepted 536 of the 2,288 sources we saw in the search of step one based on relevance of the title.

Several types of sources came up after completing the first pass of the initial search. Of the 536 sources found, 47% were conference proceedings and 38% were journal papers the remaining 15% were books, technical reports, standards, presentations, and unknown. Since books are hard to locate, and are likely to present a review of literature themselves, we eliminated books from our review. We also eliminated sources that did not have any traceable bibliographic information, which only occurred in the searches of CiteSeerX. The next few phases of the search are summarized by Figure 2.

Question	Answer
1. Are there clearly identifiable metrics validation criteria? • *Does the paper contain one or more criteria that can be extracted?* • *Is this paper an examination of what should be required to make a software engineering metric valid?* • *Does this paper exhibit scientific objectivity, or is it a "position paper"? If the paper is a position paper, reject it.*	*Yes/No*
2. If the answer to 1 is No, then why should this paper be excluded from the study?	*List reasons.*
3. What are the metrics validation criteria or groups of criteria this paper describes? • *Summarize each criterion into a single bullet point.* • *What is the motivation for this criterion?* • *Is this a single criterion, or a group of criteria?* • *Do the authors indicate this criterion as being necessary for validation or a desirable property?*	*List the criteria.*
4. How is this criteria related to other criteria? • *List the related criterion and its source.* • *Explain the rationale for the conflict* o *Establish why these criteria are related.* o *Establish that these criteria indeed are indeed opposing, or the same.* o *If these criteria conflict, do these criteria just oppose each other, or are they truly mutually exclusive?* o *If these criteria are similar, are they synonyms or is there a strong enough difference in their definition to warrant a new criterion?*	*List the criterion, source, and explanation*

Fig. 1. Primary source full-text review form.

Table I. Search Engines, Queries, and Results for the First Pass

Index/Search Engine	Search String (Query)	Total Results	Reviewed Results	Researcher
Google Scholar	software AND engineering AND metrics	125,000	270	B
IEEExplore	software AND engineering AND metrics	1,492	125	B
CiteSeerX	software AND engineering AND metrics	539,029	170	B
ACM	software AND engineering AND metrics	7,640	100	B
Google Scholar	software AND metrics AND validation	50,400	510	A
IEEExplore	software AND metrics AND validation	223	223	A
ACM	software AND metrics AND validation	2,430	300	A
CiteSeerX	software AND metrics AND validation	548,575	150	B
Google Scholar	software AND metrics AND evaluation	118,000	150	B
IEEExplore	software AND metrics AND evaluation	1,836,783	100	B
CiteSeerX	software AND metrics AND evaluation	26,953	90	B
ACM	software AND metrics AND evaluation	6,898	100	A
Total		**3,263,423**	**2,288**	**A & B**

3.2.2. Second Pass: Confirming the Abstracts. After confirming the titles, we gathered the abstracts for the 156 titles and each researcher voted whether the abstract was relevant or not with the revised criteria as shown in Table II. Researcher A voted yes to 46 of 156 abstracts and Researcher B voted yes to 59 of 156 abstracts. The researchers

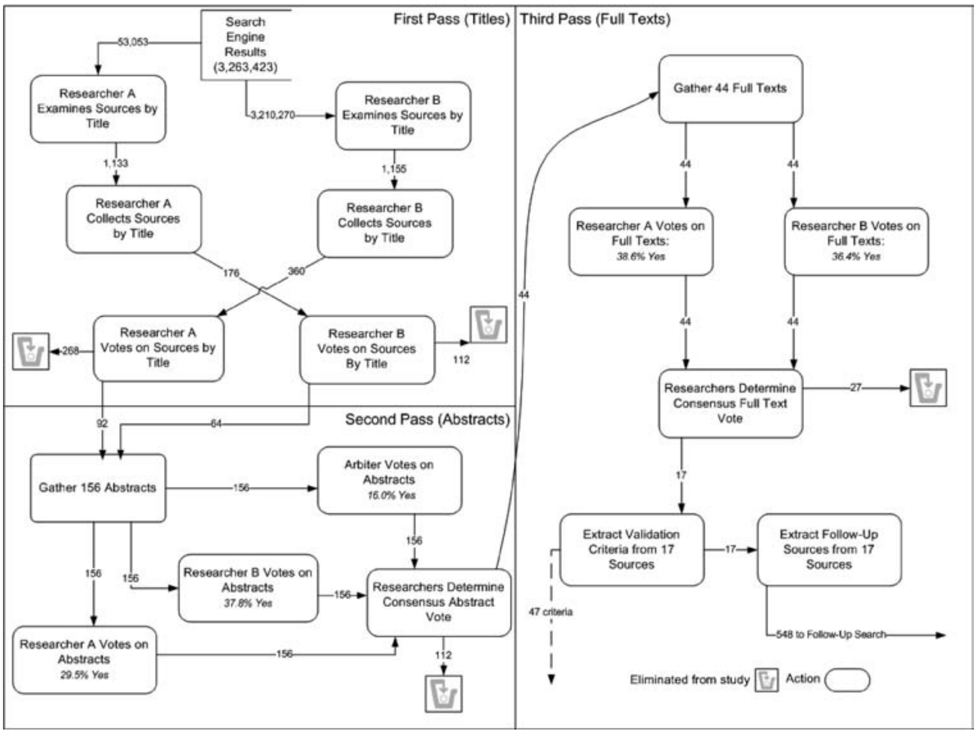
First Pass (Titles)
Third Pass (Full Texts)
Second Pass (Abstracts)
Search Engine Results (3,263,423)
53,053
3,210,270
Researcher A Examines Sources by Title
Researcher B Examines Sources by Title
1,133
1,155
Researcher A Collects Sources by Title
Researcher B Collects Sources by Title
176
360
Researcher A Votes on Sources by Title
Researcher B Votes on Sources By Title
268
112
92
64
Gather 156 Abstracts
156
Arbiter Votes on Abstracts 16.0% Yes
Researcher B Votes on Abstracts 37.8% Yes
Researcher A Votes on Abstracts 29.5% Yes
Researchers Determine Consensus Abstract Vote
112
44
Gather 44 Full Texts
Researcher A Votes on Full Texts: 38.6% Yes
Researcher B Votes on Full Texts: 36.4% Yes
Researchers Determine Consensus Full Text Vote
27
17
Extract Validation Criteria from 17 Sources
Extract Follow-Up Sources from 17 Sources
47 criteria
548 to Follow-Up Search
Eliminated from study
Action

Fig. 2. Overall view of the source-gathering procedure in the initial literature review.

Table II. Second Pass Voting Results

Voter	Keep	Discard
Researcher A	46	110
Researcher B	59	97
Overall Consensus	51	105

Table III. Second Pass Voting with Arbiter Results

Voter	Keep	Discard
Researchers A & B	51	105
Arbiter	38	118
Overall Consensus	44	112

agreed on 102 abstracts and disagreed on 54. We held a meeting to form consensus on the 54 contested abstracts. Of the 54, we voted 40 abstracts as irrelevant and 14 as relevant. This result, combined with the other consensus obtained by both researchers independently agreeing on the status of the abstracts resulted in 51 relevant abstracts total and 105 irrelevant abstracts.

The overall consensus on the abstracts, as determined previously, was passed on to the Arbiter, the third author in this paper. As shown in Table III, the Arbiter voted on the relevance of the 156 abstracts and indicated that 38 were relevant and 118 were irrelevant. The Arbiter agreed with the researchers' consensus for 121 abstracts and disagreed with the researchers on 35 abstracts. Another meeting was held to form overall consensus on the abstracts. Out of the 35 abstracts discussed at the meeting, the full research team (consisting of Researcher A, Researcher B, and the Arbiter) accepted

Table IV. Full Text Third Pass Voting Results

Voter	Yes (Keep)	No (Discard)
Researcher A	17	27
Researcher B	16	28
Agreed	29	15
Overall Final Consensus	17	27

17 abstracts and rejected 18 abstracts. We accepted 44 abstracts total (rejecting 112), and proceeded to the next phase of the review.

3.2.3. Third Pass: Full Texts. The next phase of the search was to vote on the full text for each of the 44 sources whose abstract the research team agreed upon. The voting results of the full-text pass are shown in Table IV. Researcher A and Researcher B then voted on whether the source matched the criteria for the systematic literature review by completing the form shown in Figure 1 for each source.

In this phase, the researchers agreed on 29 full texts, and disagreed on 15. The researchers held another meeting to discuss the full list of all 44 full texts. After forming consensus, 27 full texts were rejected, and 17 were accepted (note: we added three sources in the follow-up search described in Section 3.3). The notes gathered during the establishment of the chosen sources were used to develop the full list of 47 metrics validation criteria described in Section 4.

In summary, we searched a population of over 3 million sources, examined 2,288 titles, and selected 536 sources based on the relevance to our research objective. Of those 536 sources, we voted to confirm the relevance and were left with 156 relevant titles. Those 156 titles were confirmed for their abstract content by each researcher, and then the arbiter. We formed consensus on 44 relevant abstracts. The researchers then used the full texts of each of the 44 sources to decide on a consensus of 17 sources that were used to construct this review, and are listed in the next section.

3.3. Conducting the Follow-Up Search

After extracting data from the final 17 primary sources from the first search, we determined that in the interest of thoroughness we should conduct a second search based upon the references of our sources. The 17 sources for our study contained 548 unique references, which we used to conduct what is referred to as the “follow-up search”. The titles were easier to exclude in the follow-up review because the research questions had been more solidly formulated, and the researchers had experience with knowing what papers were needed for the study. The pass through the titles resulted in 22 publications. We then examined the abstracts for those 22 publications and found eight relevant sources. After reviewing the full text for these eight sources, we selected three sources that were relevant, bringing the total number of papers included in our study to 20. None of the sources from the follow-up review contained new metrics validation criteria. We did, however, find additional support for metrics validation criteria we had already discovered.

4. EXTRACTING THE VALIDATION CRITERIA

We analyzed each of the 20 sources and recorded all of the validation criteria presented by each source. After extracting all criteria from all sources, we combined identical criteria together, resulting in a list of 47 unique criteria.

Our intention is not to indicate to the reader that validating a metric requires the satisfaction of all 47 validation criteria, nor that the reader can freely select which criteria apply to a specific situation. Rather, we intend the list presented in Figure 3 to act as (i) a reference point for this article and for future research; and (ii) an invitation

1. A priori validity
2. Actionability
3. Appropriate Continuity
4. Appropriate Granularity
5. Association
6. Attribute validity
7. Causal model validity
8. Causal relationship validity
9. Content validity
10. Construct validity
11. Constructiveness
12. Definition validity
13. Discriminative power
14. Dimensional consistency
15. Economic productivity
16. Empirical validity
17. External validity
18. Factor independence
19. Improvement validity
20. Instrument validity
21. Increasing growth validity
22. Interaction sensitivity
23. Internal consistency
24. Internal validity
25. Monotonicity
26. Metric Reliability
27. Non-collinearity
28. Non-exploitability
29. Non-uniformity
30. Notation validity
31. Permutation validity
32. Predictability
33. Prediction system validity
34. Process or Product Relevance
35. Protocol validity
36. Rank Consistency
37. Renaming insensitivity
38. Repeatability
39. Representation condition
40. Scale validity
41. Stability
42. Theoretical validity
43. Trackability
44. Transformation invariance
45. Underlying theory validity
46. Unit validity
47. Usability

Fig. 3. List of all 47 validation criteria found in the review.

the research community to continue the discussion of what should be required to designate a metric as valid. Therefore, this list is not conflict-free (i.e. some criteria contradict each other).

We first summarize the 47 criteria in Figure 3 in alphabetical order. We further synthesize the resultant criteria in Section 5.

For the rest of this article, a reference to a number with a pound sign denotes a reference to one of these criteria (e.g., #30 refers to predictability). In Table V we present a matrix of the criteria by their numbers (see Figure 3), and the originating sources.

In the following list, the sentence after the name, in italics, is our definition of the criterion, as combined from each of the cited papers. Then, after the definition is a brief discussion of that specific criterion.

(1) *A priori validity*. A metric has a priori validity if the attributes in association are specified in advance of finding a correlation [Fenton 1994; Courtney and Gustafson 1993; Briand et al. 1995; Baker et al. 1990]. A priori validity is often referred to in the converse as a “shotgun correlation” [Courtney and Gustafson 1993] where a correlation between a metric and a quality factor is found in a data set, then explained post hoc. If one examines enough metrics (discarding any lack of correlation), one could eventually find a statistically significant, yet fortuitous correlation. Instead, the authors point out that the hypothesis of a metric’s meaning ought to precede finding a correlation.

(2) *Actionability*. A metric has actionability if it allows a software manager to make an empirically informed decision based on the software product’s status [Fenton

Table V. Matrix of Criteria and Their Cited Source

	(Fenton 1994)	(Kitchenham, Pfleeger et al. 1995)	(Curtis 1980)	(Roche 1994)	(Weyuker 1988)	(Jones 1994)	(Schneidewind 1991)	(El Emam 2000)	(Fenton and Neil 1999)	(Briand, Emam et al. 1995)	(Briand, Emam et al. 1995)	(Henderson-Sellers 1996)	(Harman and Clark 2004)	(Cavano and McCall 1978)	(Lincke and Lowe 2006)	(Baker, Bieman et al. 1990)	(Courtney and Gustafson 1993)	(Bush and Fenton 1990)	(Fenton 1991)
# 1	X										X					X	X		
# 2				X					X										
# 3		X																	
# 4		X			X														
# 5	X						X												
# 6		X														X			
# 7									X										
# 8			X	X															
# 9			X																
# 10			X																
# 11														X					
# 12				X				X						X	X			X	
# 13							X												
# 14		X										X							
# 15						X													
# 16																			
# 17	X							X			X					X			
# 18				X															
# 19								X											
# 20		X																	
# 21					X														
# 22					X														
# 23			X																
# 24								X								X			
# 25					X														
# 26			X																
# 27								X											
# 28														X					
# 29					X														
# 30												X							
# 31					X														
# 32	X		X	X			X	X										X	X
# 33								X								X			X
# 34				X			X			X									
# 35		X																	
# 36							X												
# 37					X														

(Continued)

Table V. (Continued)

	(Fenton 1994)	(Kitchenham, Pfleeger et al. 1995)	(Curtis 1980)	(Roche 1994)	(Weyuker 1988)	(Jones 1994)	(Schneidewind 1991)	(El Emam 2000)	(Fenton and Neil 1999)	(Briand, Emam et al. 1995)	(Briand, Emam et al. 1995)	(Henderson-Sellers 1996)	(Harman and Clark 2004)	(Cavano and McCall 1978)	(Lincke and Lowe 2006)	(Baker, Bieman et al. 1990)	(Courtney and Gustafson 1993)	(Bush and Fenton 1990)	(Fenton 1991)
# 38							X	X											
# 39	X	X											X			X		X	X
# 40	X	X						X		X									X
# 41			X					X						X					
# 42		X								X	X								
# 43							X												
# 44													X						
# 45		X		X				X								X			
# 46	X	X																	
# 47														X					

and Neil 2000; Roche 1994]. The metric should reflect some property of the software in a way that enables managers to make decisions during the software development lifecycle. Roche uses the terms "interpretative guidelines" and "recommendations for action" when discussing the actionability of a metric. We introduce the term "actionability" as our interpretation of the idea discussed by the authors.

(3) *Appropriate Continuity*. A metric has appropriate continuity if the metric is defined(or undefined) for all values according to the attribute being measured [Kitchenham et al. 1995]. Kitchenham et al. phrase this criterion as "the metric should not exhibit any unexpected discontinuities." This issue arises typically from fraction calculations where the denominator can be zero. For example, if one used the metric F/LOC (faults per line of code), one would need to define F/LOC for LOC = 0, since the equation is discontinuous.

(4) *Appropriate Granularity*. A metric has appropriate granularity if the mapping from attribute to metric is not too finely- or coarsely-grained [Kitchenham et al. 1995; Weyuker 1988;]. This criterion is a grouping ofthree of Weyuker's complexity criteria. Kitchenham et al. also discuss these three criteria, but in terms of all metrics (not just complexity). The three criteria could be grouped as fine, medium, and coarse granularity.

(a) A metric has fine granularity if there are only finitely many programs that can achieve agiven measure Weyuker's Property 2. For example, the cyclomatic number is not finely grained as one can create an infinite number of programs that have the same cyclomatic number.

(b) A metric has medium granularity if there are two programs that compute the same function, but have different measures Weyuker's Property 4. This property is based on the idea that different programs can perform identical functionality with differing implementations and, therefore, result in different complexities. For example, cyclomatic complexity has medium granularity because one can write two programs that have different complexities, but still perform the same functionality.

(c) A metric has coarse granularity if two different programs can result in the same measure Weyuker's Property 3. That is, not every measurement needs to be unique to a specific program. The two programs can represent two different implementations altogether (not just renamings of each other) and can still have the same complexity value.

(5) *Association* A metric has association validity if it has a direct, linear statistical correlation with an external quality factor [Schneidewind 1991, 1992; Fenton 1994]. We use the term "direct" to imply that the metric is measured without the use of a model (e.g., regression). Measurement of this criterion is typically done by the Pearson correlation coefficient or the coefficient of determination (e.g., R^2) in a linear regression. Fenton uses the term "external correlation" when discussing association validity. Note that Schneidewind and Fenton explicitly differentiate this term from "prediction" validity (#32). Association validity differs from predictability in that association does not separate training sets from test sets (e.g., using cross-validation), nor does association involve using a model.

(6) *Attribute validity*. A metric has attribute validity if the measurements correctly exhibit the attribute that the metric is intending to measure [Kitchenham et al. 1995; Baker et al. 1990]. Kitchenham's discussion of attribute validity focuses on the actual measurements of a metric. For example, if one were measuring the attribute validity of the "psychological complexity" of code, one might poll a group of developers and empirically evaluate their agreement. Kitchenham makes the point that the attributes being measured need to be aspects of software that have both intuitive and well-understood meanings.

(7) *Causal model validity*. A metric has causal model validity if it can be used in a causal model that explains a quality factor [Fenton and Neil 2000]. If a metric can be used as a variable in a model of causality (e.g., Bayesian Belief Networks), then more credence can be given to the metric's ability to cause changes in an external quality factor. Note that a metric functioning as a variable within a causal model does not imply a causal relationship, but indicates a stronger possibility of a causal relationship. We note the distinction between causal model validity and causal relationship validity (see #8).

(8) *Causal relationship validity*. A metric has causal relationship validity if it has a causal relationship to an external quality factor [Roche 1994; Curtis 1980]. Rather than having only a statistical correlation with an external quality factor, the attribute measured by a metric must be shown to *cause* changes in the quality attribute by a properly designed and controlled experiment. For example, warnings from a perfect static analyzer that finds null dereferences could be used as a metric to predict null reference failures in a system because executing a fault *causes* a failure. Causal relationship validity is different from causal model validity (#7) because a causal model does have to dictate a relationship.

(9) *Content validity*. A metric has content validity if it applies "thoroughly to the entire domain of interest" [Curtis 1980]. A metric must capture the *entire* notion of an attribute to be considered valid. For example, McCabe's cyclomatic complexity number might be considered content invalid in the domain of "complexity" as it does not account for psychological complexity because obfuscating variable names does not affect cyclomatic complexity but does affect psychological complexity. Curtis uses the term "face valid" to refer to a metric broadly sampling a domain. Interestingly, we did not encounter an example that the authors presented that had content validity.

(10) *Construct validity*. A metric has construct validity if the gathering of a metric's measurements is suitable for the definition of the targeted attribute. [Curtis 1980]. The word "construct" in this sense refers to the tool, instrument, or procedure used to gather measurements. Curtis refers to construct validity as when "the operational definition yields data related to an abstract concept". Showing that a metric does not have construct validity means showing that a specific implementation of a metric is not valid.

(11) *Constructiveness*. A metric is constructive if it helps the researcher understand software quality [Cavano and McCall 1978]. For example, if a metric measures the attribute of "size", but is not correlated with quality (e.g., there are an equal number of high quality large components, and low quality small components), then the metric is not constructive.

(12) *Definition Validity*. A metric has definition validity if the metric definition is clear and unambiguous such that its collection can be implemented in a unique, deterministic way [Lincke and Lowe 2006; Cavano and McCall 1978; Roche 1994; Bush and Fenton 1990; El Emam 2000]. We use the term "deterministic" from Lincke et al. to mean that, given a definition of a metric, the measurement made would be consistent across all who implement the definition, implying full clarity and a lack of ambiguity. The term "unique" implies that, given a definition and an artifact, the metric result should be unique from all other measurements. Cavano calls this kind of definition "consistent" and "detailed"; Roche calls it "clear" and "unambiguous"; Bush calls this a "precise" definition.

(13) *Discriminative Power*. A metric has discriminative power if it can show a difference between high-quality and low-quality components by examining components above/below a predetermined critical value [Schneidewind 1991, 1992]. The discriminative power criterion is used to establish sets of metric values that should be considered "too dangerous" or "off limits". For example, if the LOC metric has discriminative power for number of faults with a critical value of 100, then components with over 100 lines of code are more likely to have a dangerous number of faults than files with fewer than 100 lines of code.

(14) *Dimensional Consistency*. A metric has dimensional consistency if the formulation of multiple metrics into a composite metric is performed by a scientifically well-understood mathematical function [Kitchenham et al. 1995; Henderson-Sellers 1996]. For example, converting from a vector to a scalar loses vital information about the entity. Kitchenham uses the example of multiplying the X,Y position coordinates on a Cartesian plane is meaningless because, while we would obtain a value, we would not know what attribute is being measured.

(15) *Economic Productivity*. A metric has economic productivity if using the metric quantifies a relationship between cost and benefit [Jones 1994]. That is, a metric is considered invalid if it does not result in saving money in the long run. Jones stipulates that gathering and collecting a metric must not be cost-prohibitive. Furthermore, achieving "good" scores of the metric must not also be cost-prohibitive. For example, removing all known faults in a software system may be cost-prohibitive, even if it would improve a "number of known faults" metric. We introduce the term "economic productivity" as an interpretation of Jones' arguments against commonly used metrics.

(16) *Empirical validity*. A metric has empirical validity if experimentation and/or observation corroborates either (i) the intended measurement of a metric; or (ii) the relationship between the metric and an external software quality factor. Empirical validation is typically performed with statistical analysis of data obtained via experiments or project history.

(17) *External validity*. A metric has external validity if it is related in some way (e.g., by prediction, association or causality) with an external quality factor [El Emam 2000; Briand et al. 1995; Baker, et al. 1990; Fenton 1994]. External validity is a broad category of validation criteria. For example, if one showed that files with a high LOC is associated with having many faults, then LOC would be considered externally valid. However, correlating LOC with another internal metric, such as code churn, would not be considered external validation. El Emam equates the term “external” with “empirical” validation, and we distinguish the two terms. For an in-depth discussion on the difference, see Section 5.1.

(18) *Factor independence*. A metric has factor independence if the individual measurements used in the metric formulation are independent of each other [Roche 1994]. Roche describes factor independence as an “analytical principle” that applies when a metric is composed of several, individual measurements.

(19) *Improvement validity*. A metric has improvement validity if the metric is an improvement over existing metrics [El Emam 2000]. The term “improvement” that El Emam uses denotes a broad range of possible improvements that a metric could have. Examples of improvements include ease of measurement, stronger association with a quality factor, or a closer representation to the attribute being measured.

(20) *Instrument validity*. A metric has instrument validity if the underlying measurement instrument is valid and properly calibrated [Kitchenham et al. 1995]. For example, consider a tool has been improperly implemented the definition for branch coverage; this tool, as an instrument, would be invalid. As a result, the data gathered from that tool (in this case, the poorly defined branch coverage) would also be considered invalid.

(21) *Increasing growth validity*. A metric has increasing growth validity if the metric increases when concatenating two entities together [Weyuker 1988]. Said another way, a metric should never go down if one is concatenating code together. Note that this criterion requires a specific scale type, which is why Kitchenham et al. explicitly reject this criterion. For example, the code coverage for different program bodies should never decrease by concatenating two components together.

(22) *Interaction sensitivity*. A metric has interaction sensitivity if two components of a program result in different metrics depending on how they interact with one another [Weyuker 1988]. Weyuker’s Property 6 uses this criterion to discuss how components interact with one another, which can be specific to complexity, but might be a property that applies to other software metrics. For example, cyclomatic number is not concatenation sensitive as the cyclomatic number of a concatenation of program bodies is *always* equal to the sum of the individual cyclomatic numbers of each body. In other words, cyclomatic number does not take into account the interaction between components, only individual components. The term “concatenation sensitive” is not used explicitly by Weyuker, but is our interpretation based on the author’s mathematical definition.

(23) *Internal consistency*. A metric has internal consistency if “all of the elementary measurements of a metric are assessing the same construct and are inter-related” [Curtis 1980]. Curtis also argues that a lack of internal consistency results in losing the ability to interpret the metric.

(24) *Internal validity*. A metric has internal validity if the metric correctly measures the attribute it purports to measure [El Emam Baker et al. 1990]. Internal validity is a broad category of validation criteria that is solely concerned with the metric itself, regardless of being associated with an external quality factor. Most authors of our sources discuss some form of internal validity, however, many will

use the term "theoretical" synonymously with "internal". For a deeper discussion on this issue, see Section 5.1.

(25) *Monotonicity*. A metric has monotonicity if the components of a program are no more complex than the entire program [Weyuker 1988]. For example, if a module has one highly complex method, then the entire module should be at least as complex as that method. This notion applies to other metrics as well, not just complexity metrics.

(26) *Metric reliability*. A metric has reliability if the measurements are "accurate and repeatable" [Curtis 1980]. Curtis argues that a "reliable" metric ought have little random error. Curtis explicitly states that metric reliability is a super-category for internal consistency (#23) and stability (#39).

(27) *Noncollinearity*. A metric has noncollinearity if it is still correlated with an external quality factor after controlling for confounding factors [El Emam 2000]. For example, if a complexity metric was highly influenced by code size to the point where no extra variance is explained once code size was included, then the complexity metric is collinear (and therefore does not pass this criterion).

(28) *Nonexploitability*. A metric exhibits nonexploitability if developers cannot manipulate a metric to obtain desired results [Cavano and McCall 1978]. We introduce use the term "exploitability" to describe the phenomenon where people can manipulate a metric's measurements without changing the attribute being measured. For example, if LOC is being used as an effort metric, then a developer could start writing exceedingly verbose code to look more productive. Cavano uses the term "stability" when referring to nonexploitability, which is not to be confused with our use of stability (#41).

(29) *Nonuniformity*. A metric has nonuniformity if it can produce different values for at least two different entities [Weyuker 1988]. As Weyuker states, "A metric which rates all programs as equal is not really a measure".

(30) *Notation validity*. A metric has notation validity if the metric is reasoned about "mathematically with precise, consistent notation" [Henderson-Sellers 1996]. Henderson-Sellers argues that a metric cannot be validated by other researchers if the metric is not properly defined with correct, consistent, and unambiguous mathematical notation. Furthermore, a metric with an unclear definition could mislead other researchers to draw wrong conclusions about a metric.

(31) *Permutation validity*. A metric has permutation validity if the metric values are responsive to the order of the statements. [Weyuker 1988]. Weyuker argues this criterion for complexity. Her argument is that the interaction of statements in a program affects the notion of complexity, so permuting the statements in a program ought to affect the complexity of a program.

(32) *Predictability*. A metric has predictability if it can be shown to predict values of an external quality factor with an acceptable level of accuracy [Fenton 1994; Fenton 1991; Schneidewind 1991; Schneidewind 1992; Curtis 1980; Roche 1994; Bush and Fenton 1990; El Emam 2000]. Most of the authors in our sources allude to some form of prediction as one way to validate a metric with an external quality factor. In each discussion, the authors imply that showing a metric to be predictive implies that, historically, a metric could have been used to assess quality in the system. The "acceptable" level of accuracy would change from process to process and must be interpreted according to the domain of interest. Note that the Fenton and Schneidewind specifically differentiate predictability from association (#5).

(33) *Prediction System validity*. A metric has prediction system validity if the metric is part of an model with procedures on how to use the model, which both must be specified before the study takes place [Baker et al. 1990; Fenton and Kitchenham

1991; El Emam 2000]. The authors define a prediction system as involving "a mathematical model and prediction procedures for it". A prediction system has metrics in addition to models, and a prediction system can be validated. However, the authors emphatically stress that a metric does not always need to be part of a prediction system to be validated.

(34) *Process or Product Relevance*. A metric has product or process relevance if it can be "tailored to specific products or processes" [Roche 1994; Briand et al. 1996; Schneidewind 1991, 1992]. Roche argues that metrics validated in one domain ought to be transferable to other domains.

(35) *Protocol validity*. A metric has protocol validity if it is measured by a widely accepted measurement protocol [Kitchenham 1995]. The example that Kitchenham et al. provide is measuring a person's height: the agreed-upon protocol is from the feet to the head, and not including an upstretched arm.

(36) *Rank Consistency*. A metric has rank consistency if it shares the same ranking as a quality factor [Schneidewind 1991; Schneidewind 1992]. For example, if code churn were to have rank consistency with number of faults, then the ranking of files by code churn would match the ranking of files by number of faults. The rank consistency validation criterion is meant for direct relationships.

(37) *Renaming Insensitivity*. A metric has renaming insensitivity if renaming parts of a program does not affect the metric's measurement [Weyuker 1988]. For example, if one were to rename all of the variables in a program, the complexity value should not change. Although Weyuker specified this property for complexity, the metric can be generalized to other metrics.

(38) *Repeatability*. A metric has repeatability if the metric is shown to be empirically valid for multiple different projects or throughout the lifetime of one projects [Schneidewind 1991, 1992; El Emam 2000] Schneidewind defines the repeatability criterion so that one cannot make a claim of validation on simply one or two project case studies. While Schneidewind does not provide a concrete number, he implies that the number of projects required to obtain repeatability might differ from person to person depending on what they use the metric for. El Emam states, "only when evidence has accumulated that a particular metric is valid across systems and across organizations can we draw general conclusions."

(39) *Representation Condition*. A metric satisfies the Representation Condition if the attribute is a numerical characterization that preserves properties of both the attribute and the number system it maps to [Fenton 1991; Kitchenham et al. 1995; Fenton 1994; Bush and Fenton 1990; Harman and Clark 2004; Baker et al. 1990]. Under the Representation Condition, any property of the number system must appropriately map to a property of the attribute being measured (and vice versa). Fenton [1994] describes the Representation Condition as a two-way correspondence between a metric and an attribute. Kitchenham et al. [1995] state that "intuitive understanding" of an attribute is preserved when mapping to a numerical system. All of the authors cite the Representation Condition from Measurement Theory, a scientific discipline not specific to software engineering.

(40) *Scale validity*. A metric has scale validity if it is defined on an explicit, appropriate scale such that all meaningful transformations of the metric are admissible [Briand et al. 1996; Fenton 1991; Kitchenham, Pfleeger et al. 1995; Fenton 1994; El Emam 2000]. The scales discussed are typically: nominal, ordinal, interval, ratio, and absolute. Each scale type denotes a specific set of transformations that dictate how the metric can be used. As one example, adding two values of a metric of nominal scale "Yes/No" is not admissible. As another example, the temperature Fahrenheit is of interval scale which has subtraction as an admissible transformation, meaning that the difference in temperature between

50 to 60 degrees and 60 to 70 degree are the same. Ratios, however, are not admissible on the interval scale, meaning that 50 degrees is not twice as hot as 25 degrees. Many of the sources also denote specific statistical tests that ought to be run in validating metrics of different scale types.

(41) *Stability*. A metric has stability if it produces the same values "on repeated collections of data under similar circumstances" [El Emam 2000; Curtis 1980; Cavano and McCall 1978]. One example of a metric that might not be fully stable is the number of failures in a system. Since the existence of a failure is ultimately a decision made by humans (especially in the case of validation failures), two humans may disagree on whether specific system behavior is a failure or not due to an ambiguous requirements specification. Cavano calls this "fidelity".

(42) *Theoretical validity*. A metric has theoretical validity if it does not violate any necessary properties of the attribute being measured [Kitchenham et al. 1995; Briand 1996, 1995]. Theoretical validity is a broad category of validation criteria that involve making arguments about the properties of a metric. Theoretical validation is usually performed using formal logic. Many authors use the term "theoretical" synonymous with "internal" validity (#24), however we differentiate the two. For a deeper discussion on this issue, see Section 5.1.

(43) *Trackability.* A metric has trackability if the metric changes as the external quality factor changes over time [Schneidewind 1991; Schneidewind 1992]. A metric should reflect the external quality factor in such a way that if the external quality factor changes, then the metric also changes in the same direction. For example, if a high cyclomatic number is shown to be trackable with the number of faults, then reducing the number of faults in a program (e.g., fixing the faults) should, in turn, reduce the complexity. Schneidewind discusses using trackability to assess whether a component is improving, degrading, or stagnating in quality over time.

(44) *Transformation Invariance*. A metric has transformation invariance if it results in the same measurement for two semantically-equivalent programs (Harman and Clark 2004). We use the term "semantically equivalent" to mean that a compiler could potentially interpret two syntactically-different programs as the same. Harman et al. claim that if one were to make semantics-preserving changes to a program, then the value of the metric should not change. Weyuker's Renaming (#36) is one special case of semantic equivalence. However, two different implementations with indistinguishable results are not necessarily semantically equivalent. Note that the term "transformation" here is not to be confused with the admissible transformations mentioned in scale validity (#38); admissible transformations are transformations of numbers, and this criterion refers to transforming a program.

(45) *Underlying theory validity*. A metric has underlying theory validity if it its construction is based upon an underlying theory that has validity within the domain of the application [El Emam 2000; Roche 1994; Kitchenham et al. 1995; Baker et al. 1990]. To consider a metric to be valid, there must be a valid theory that describes how the metric measures what it is supposed to measure. For example, the underlying theory behind code churn is that if code has been shown to change substantially in the version control system, then the project itself has undergone a substantial amount of change. We note here that underlying theory validity applies to both internal and external validity; but the difference in these cases is in how the theory is used. On the internal side, the underlying theory required for validity must state why the metric in question is worth measuring or why the metric is an accurate representation of the attribute being measured from a theoretical standpoint. On the external side, the underlying theory is why a metric would be statistically associated with an external software quality factor (or external attribute).

(46) *Unit validity*. A metric has unit validity if the units used are an appropriate means of measuring the attribute [Kitchenham et al. 1995; Fenton 1994]. For example, fault rate may be used to measure program correctness or test case effectiveness [Kitchenham et al. 1995].

(47) *Usability*. A metric has usability if it can be cost-effectively implemented in a quality assurance program [Cavano and McCall 1978]. A metric must be feasibly collected within a process. For example, a metric that requires several months of computation might not be considered usable.

5. MAPPING THE VALIDATION CRITERIA

The diversity of metrics validation criteria presented in Section 4 indicates that software engineering metrics researchers have not converged on what constitutes a valid metric. In this section, we describe how we mapped our criteria into a single categorization.

Our approach to mapping the validation criteria was a bottom-up process. We first noticed several similarities and differences amongst the criteria. For example, many criteria dealt with relating to a quality factor, while others dealt with the meaning of a metric. Whenever we found criteria with similarities, we grouped the criteria together. In other cases, we noticed that some validation criteria are not atomically satisfiable criteria, but broad categories that can contain many criteria. For example, internal validity (#24) cannot be atomically satisfied because it is a broad category of criteria. Whenever we determined that one criterion was a specific instance of a separate, broader criterion, we marked the broad criterion as a category. A specific criterion, conversely, is presented as atomic and concretely satisfiable by its sources.

In the mapping process, we found that all of our criteria groupings could be described by categories we had already discovered. Thus, we did not introduce new categories of validation criteria, we used only the categories referred to in the literature in our mapping process.

The result of our mapping process was a categorization of the 47 criteria, as presented in Figure 4. An arrow between a criterion and a category indicates that the criterion is a part of or specific instance of that group. Specific criteria are represented as boxes, categories of criteria are represented as ovals. The number in parentheses refers to the number of references that directly discussed the criterion.

Our mapping represents the validation criteria and categories we found in the literature. We do not view this mapping to be a complete list of all possible criteria. Thus, for any given category, a criterion may be presented in the future. For example, (shown in Figure 4) attribute validity may not be the only kind of empirical, internal validity, but is the only kind we have come across in our review. Additionally, empirical (#16) and theoretical (#42) required multiple categorizations which are described in detail in Section 5.1.

In the following sections, we describe our reasoning behind some of the major decisions we made in the mapping process.

5.1. Internal/External vs. Theoretical/Empirical

Beneath our top-level categories are theoretical validity (#42), and empirical validity (#16). While some authors equate the terms “internal” with “theoretical” and “empirical” with “external”, we distinguish the ideas. We view internal and external validation as dealing with what is being validated, while theoretical and empirical validation deal with how a metric is validated. Specifically, internal validation deals with how well a metric measures an attribute, whereas as external validation relates a metric to a quality factor (i.e., another metric). Theoretical validation uses logic to argue formally whether a metric is valid or not, while empirical validation employs analysis of data

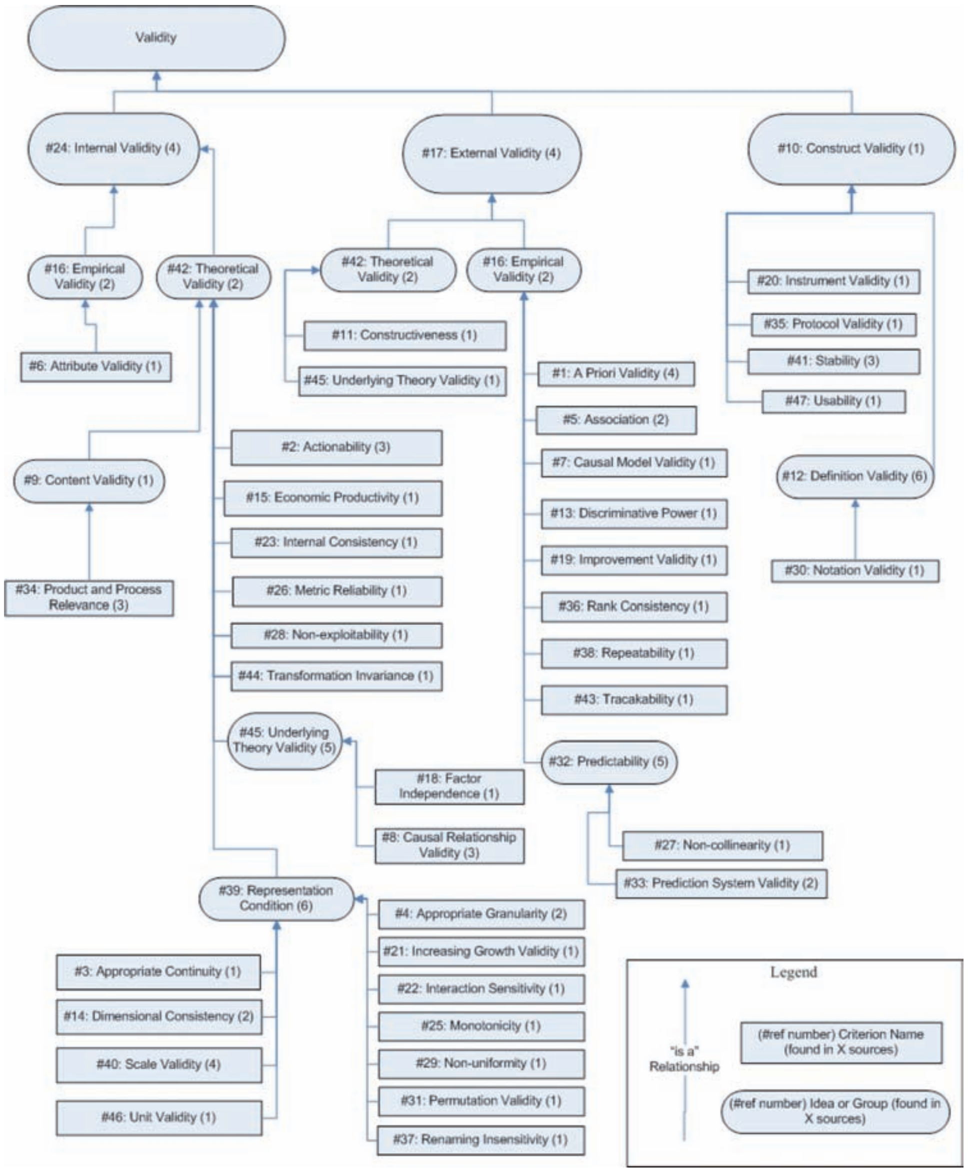
Validity
#24: Internal Validity (4)
#17: External Validity (4)
#10: Construct Validity (1)
#16: Empirical Validity (2)
#42: Theoretical Validity (2)
#42: Theoretical Validity (2)
#16: Empirical Validity (2)
#6: Attribute Validity (1)
#11: Constructiveness (1)
#45: Underlying Theory Validity (1)
#20: Instrument Validity (1)
#35: Protocol Validity (1)
#41: Stability (3)
#47: Usability (1)
#1: A Priori Validity (4)
#5: Association (2)
#7: Causal Model Validity (1)
#13: Discriminative Power (1)
#19: Improvement Validity (1)
#36: Rank Consistency (1)
#38: Repeatability (1)
#43: Tracakability (1)
#12: Definition Validity (6)
#30: Notation Validity (1)
#9: Content Validity (1)
#34: Product and Process Relevance (3)
#2: Actionability (3)
#15: Economic Productivity (1)
#23: Internal Consistency (1)
#26: Metric Reliability (1)
#28: Non-exploitability (1)
#44: Transformation Invariance (1)
#45: Underlying Theory Validity (5)
#18: Factor Independence (1)
#8: Causal Relationship Validity (3)
#32: Predictability (5)
#27: Non-collinearity (1)
#33: Prediction System Validity (2)
#39: Representation Condition (6)
#3: Appropriate Continuity (1)
#14: Dimensional Consistency (2)
#40: Scale Validity (4)
#46: Unit Validity (1)
#4: Appropriate Granularity (2)
#21: Increasing Growth Validity (1)
#22: Interaction Sensitivity (1)
#25: Monotonicity (1)
#29: Non-uniformity (1)
#31: Permutation Validity (1)
#37: Renaming Insensitivity (1)
Legend
"is a" Relationship
(#ref number) Criterion Name (found in X sources)
(#ref number) Idea or Group (found in X sources)

Fig. 4. The 47 criteria, categorized with number of references in parenthesis.

from experimentation or observation. The authors equate the terms because, as we have seen in our review, researchers typically perform internal validation theoretically and external validation empirically. One exception, however, is attribute validity (#6) which is an empirical way to validate a metric internally.

The notion of an underlying theory (#45) appears in both the external, empirical and internal, theoretical categories. In each situation, the actual theory would be different. An underlying theory for internal validation would be about how a metric measures a given attribute, whereas the underlying theory in external validation would be about how one attribute relates to another attribute.

5.2. Construct Validity

Construct validity (#10) differs from internal and external in that it deals with how a metric is implemented and studied (i.e., the construct itself). The construct validity of a metric will differ from research project to research project. For example, one could have a metric that was internally valid and was externally valid, but the analysis may be wrong because of an incorrectly implemented algorithm (#20).

Definition validity (#12) provides strong motivation for a separate category from internal (#24) and external (#17) validity. Definition validity deals with providing a clear, unambiguous, deterministically-defined metric. Many authors [Lincke and Lowe 2006; Henderson-Sellers 1996; Baker et al. 1990; El Emam 2000; Roche 1994] stress the importance of providing clear definitions for both the metrics and the analysis in metrics validation. Providing a valid metric definition (e.g., in a publication) allows that metric to be correctly reproduced so that it can be correctly studied and used by other researchers.

5.3. Representation Condition and Internal, Theoretical Validity

The Representation Condition (#39) is a mathematical property taken from Measurement Theory that deals with the relationship between a metric's attribute and the number system it maps to. Some authors [Fenton 1994; Kitchenham, Pfleeger et al. 1995; Baker et al. 1990] claim that satisfying the Representation Condition is the *same* as internal, theoretical validity. However, we found several criteria that are not concerned directly with Representation Condition, but are internal and theoretical. For example, actionability (#2) can be satisfied by making a logical argument about the attribute being measured, but does not deal with the mathematical mapping between an attribute and the number system.

5.4. Prediction, Causation, and External Validity

The external, empirical validity category included all of the criteria that involved some sort of statistical analysis with a quality factor. Some of the authors [Baker et al. 1990; Fenton 1994; Schneidewind 1992] point out that external, empirical validation is not made up entirely of prediction. Of Schneidewind's six validation criteria [Schneidewind 1992] (all of which fall into external, empirical), the notion of prediction is its own category (#30).

Schneidewind's trackability validation criterion (#43) is another example of an external criterion not related to prediction. Trackability bears closest resemblance to causal relationship validity (#8). Trackability states that a metric must change as the external quality factor changes. If a metric's attribute truly *causes* change in software's external quality, then trackability ought to be empirically supported (e.g., increasing quality reduces code size, and vice versa).

6. SPECTRUM OF PHILOSOPHIES

We postulate that the reason researchers have yet to determine a sufficient set of criteria comes from a fundamental difference in philosophies about how a metric ought to be used. A metric could be used either as a pragmatic way of improving software and its surrounding processes or more rigorously as a window into understanding the very nature of software. We view the differing perspectives as being based on two opposing philosophies.

—The *goal-driven* philosophy holds that the primary purpose of a metric is to use it in assessment, prediction, and improvement. Validating a metric internally (#24), then, only serves to improve the usefulness of the metric in its application. If a metric is "almost" internally valid (i.e., passes many internal validation criteria, but fails a few), those with a goal-driven perspective would not see a major problem as long as the project benefited from the guidance and decision support gleaned from using the metric.

—The *theory-driven* philosophy views that the primary purpose of a metric is to gain understanding of the nature of software. Validating a metric internally, then, is of central importance. If a metric does not follow the Representation Condition (#39), for example, then the metric is invalid and should not be further studied. Those of the theory-driven perspective often denounce the use of external, empirical studies claiming that the studies do not include a thorough discussion of internal, theoretical validity.

6.1. Goal-Driven Philosophy Principles

More specifically, the goal-driven philosophy can be characterized by the following ideals.

—*Specify measurement goals.* Metrics should be gathered, defined and analyzed with respect to a specific goal; that is, how to fix a given set of problems in a given project or organization. For example, Fenton [1994] and Briand et al. [1995], explain that measurement activities must always have clear objectives and in fact be objective-based (e.g., goal/question/metric Basili and Weiss [1994]) also agree with this idea.

—*Goals vary with specific processes and products.* Metrics can only be validated to a certain project, process, or environment. Schneidewind [1991] and Roche [1994] contend that metrics should be used in similar processes, products or environments.

—*Validation is a continuous process*. Since metrics are validated to specific projects, a metric that is valid today may not be valid tomorrow, and a metric that is valid on the current project may not be relevant on the next project. For example, Schneidewind explains that the fundamental problem in software metrics validation is the following: there must be a project in which metrics are validated, and a different project in which the metrics are applied. There could be significant time lags, product differences, and process differences between these two projects, and these scheduling differences should signal the need to exercise care in choosing the two projects such that the application of the metrics is appropriate [Schneidewind 1991].

—*All theoretical analysis eventually serves a goal*. Theoretical analysis, such as the Representation Condition, ought to be pursued, but only insofar as helping achieve the measurement goal. For example, Schneidewind explains that the purpose of metrics validation is to identify metrics that are related to software quality factors (which are typically the measurement goal) [Schneidewind 1991].

—*Metrics can have an ad-hoc definition.* A metric can be specific to a product, process, environment, or technology. For example, a metric that looks for the use of a specific module in the code would be an ad hoc definition as it would not have meaning outside of its project. Those of the goal-driven perspective would view ad hoc definitions acceptable as long as the metric leads to fulfilling a goal.

—*Actionable metrics are the best.* Actionable metrics that are associated with quality factors are the most desirable as they can be used to predict *and* improve the process. Un-actionable prediction is less useful, but can still fulfill specific goals and should still be pursued.

—*"Good enough" is good enough.* Statistical theory is only relevant to the extent that it helps us to measure the association or predictability of an actionable metric. To the goal-driven, a debate on whether a metric is defined over the correct scale type is less important than external validation because an improper scale type can still be used in practice and achieve effective results.

Goal-Driven
Theory-Driven

Fig. 5. The spectrum of diametrically-opposed philosophies

—*No universally applicable set of criteria exists*. All validation criteria serve to fulfill a specific goal, which can vary from project to project. The goal, then dictates which criteria apply to a given project.

6.2. Theory-Driven Philosophy Principles

The theory-driven philosophy can be characterized by the following ideals.

—*Must have a theory*. Metrics should be gathered based upon an underlying theory of how the measurement is representative of an attribute in the software. Improvement in a process-driven sense takes a back seat to improvement of our knowledge of software. For example, Briand et al. contend that internal attributes are interesting for software engineers as long as they are a part of software engineering theories.
—*Metrics should be generally applicable*. A metric must be universally defined before we can begin to apply it. For example, Lincke and Lowe [2006] contend that metric definitions should be independent of environments or project-specifics.
—*Metrics should be repeatable*. Metrics should be gathered systematically by a program or clearly specified procedure, so as to ensure that there are no errors during measurement. The goal-driven philosophy would agree with this idea, too, but only to the extent that the error is within an acceptable range to make an acceptable prediction.
—*Theoretical, internal validation is paramount*. All possible values of the metric must align with the assumptions of the attribute being measured. Insinuated in this idea is that a metric should be held to close scrutiny mathematically and analytically before it is ever tested empirically or applied to a process. Scale type, admissible transformations, and appropriate use of statistical techniques are all stressed.
—*Know the underlying reasons*. Eventually, the true validation of metrics can help us understand the underlying forces that cause software and the software development process to behave the way that they do.
—*Un-actionable is not useless*. Just because a metric is not actionable does not render it useless to understanding software. Some metrics are relevant because they help us characterize software for many purposes outside of prediction and other specific project goals. Therefore, those of theory-driven philosophy would view actionability as orthogonal to a metric's validity.
—*A universally applicable set of criteria does exist*. A universally-applicable set of criteria must exist if we are to understand and agree upon the nature of software.

We contend that the two ends of the spectrum are counter-opposed, meaning they compete with one another both in theory and in practice. However, the two philosophies are not mutually exclusive. A given researcher or practitioner may agree or disagree with some aspects of both extremes and borrow motivations from both philosophies. Even within the same sources, we frequently discovered metrics validation criteria that seemed to emanate from both philosophies. As such, we contend that the two philosophies below represent the two ends of a spectrum as shown in Figure 5.

At first glance, one may think that all practitioners are goal-driven and all researchers are theory-driven, however, we do not consider this to be the case. For example, a developer can be theory-driven because she wants a complexity metric to represent what she believes is complex code. Conversely, a researcher can be goal-driven because he cares more about the applicability of his research than being theoretically correct. Therefore, researchers and practitioners can be at any point along the spectrum.

In terms of the criteria motivation, one may consider a relative ordering of a given validation criterion or set of criteria in terms of their position on the spectrum, but we find that no objective system of measurement exists for determining such an ordering. Thus, we do not indicate a mapping of the criteria onto the spectrum we propose in this article. We do not view that one paper, author, or criterion fully adhere to one extreme or the other. However, we view the following as representative examples of criteria that are motivated by either the goal-driven philosophy or the theory-driven philosophy.

—We view five of Schneidewind's six criteria [Schneidewind 1991, 1992] to be primarily motivated by a goal-driven philosophy: association (#5), rank consistency (#36), discriminative power, (#13), predictability (#30), and trackability (#42).
—However, we view Schneidewind's sixth criterion, repeatability (#35), as being in the middle of the spectrum. If a metric has been shown to be related to a quality factor on repeated occasions, then it is reliable for satisfying specific project goals and lends credence to a universal truth.
—We view the Representation Condition (#39) and its subcriteria to be primarily motivated by a theory-driven philosophy. If a metric satisfies the Representation Condition, then understanding the number system the metric maps to implies understanding the attribute being measured.
—We view actionability (#2) to be primarily motivated by the goal-driven philosophy. Actionability is often discussed in terms of satisfying specific goals of a project [Curtis 1980; Roche 1994; Fenton and Neil 2000]. Those of the theory-driven philosophy would view actionability as orthogonal to a metric's validity: if a metric is valid but is un-actionable in practice, it still speaks to the nature of software.

7. CHOOSING APPROPRIATE METRICS VALIDATION CRITERIA

In Section 6, we described how the theory-driven and goal-driven perspectives are diametrically opposed, leading to a conflict in how the community agrees upon how a metric ought to be validated. In this section, we provide some guidance on the use of the criteria to make the information in this article more actionable. Many of the tenets of each philosophy are based on the intended use of a metric. For example, in the theory-driven perspective, intended uses of a metric involve discovering the nature of software, whereas the goal-driven perspective typically uses metrics to satisfy business goals.

One of the objectives of this review is to reveal how the software engineering community believes a metric is to be considered valid, that is, suitable and acceptable for its intended use. Introducing the intended use along with the metric, therefore, is a critical step toward metric validation. A metric that is valid for one intended use may not be valid for another intended use. For example, a metric could be useful for demonstrating universal theories about software, but also be infeasible to collect and apply in a development process. In such a case, the metric would be considered valid for the former use, and invalid for the latter use.

Deciding on a particular intended use points the researcher to specific properties of a metric that are more appropriate for that use; we call these advantages. For example, showing the metric to be a meaningful representation of the attribute is one advantage (see "Meaningfulness" in Section 7.1) and showing that it can be applied to a development process is another advantage (see "Practicality" in Section 7.1). Furthermore, showing that a metric lacks an advantage can be a way to articulate a metric's limitation. To demonstrate that a metric has specific advantages, a researcher or practitioner needs to use the criteria that provide those advantages.

In this review, we observed 11 common advantages from the 47 criteria. Each advantage is an abstract property of a metric that a researcher can demonstrate using its associated validation criteria. In Section 7.1, we describe the 11 advantages and map them to the criteria. In Section 7.2, we propose a step-by-step process of applying the validation criteria via the intended use, advantages, and the information found in previous sections of this article.

7.1. Observed Advantages of Using Various Metrics Validation Criteria

The following is a list of the advantages for metrics validation criteria that we have observed in the sources for this survey. Two of the authors of this article, Andrew Meneely and Ben Smith, examined the list of criteria to articulate the advantages of each metric validation criterion. We then synthesized our findings and collaboratively developed this list, maintaining traceability from the advantages to the criteria. We do not claim that these are the only advantages of metrics that could be demonstrated. Table VI provides our mapping from the advantages to the criteria themselves.

—*Mathematical Soundness*. This advantage enables users of the metric to safely perform specified mathematical operations on the metric without the concern that the results will misrepresent the attribute.
—*Practicality*. Software engineers can directly apply the metric to improve their software development processes and practices. Criteria that give the practicality advantage show that a metric can be feasibly deployed in a software development setting.
—*Correctness*. The implementation of a metric rarely obtains incorrect values, which applies especially to construct validity types of criteria.
—*Efficiency*. The efficiency advantage allows practitioners and researchers to save time and energy by choosing a single metric for a given measurement goal.
—*Difference-Detecting*. The difference-detecting advantage enables metrics to carefully detect differences in the attribute, while avoiding metrics that are too dependent on the measured attribute.
—*Hypothesis-Strengthening*. Empirical work gives researchers more confidence in their conclusions because of repeated observations. The hypothesis-strengthening advantage also increases researchers' confidence in the theories they present.
—*Meaningfulness*. The meaningfulness advantage implies an intuitive, easy to understand, conceptual relationship between the metric and the attribute that ensures the interpretability of the metric. Do researchers intuitively know what the metric means?
—*Decision-Informing*. The metric helps researchers and practitioners make informed decisions while developing software. For example, association may indicate that a factor is related to quality, but if the metric is unactionable it still may help with risk assessment. Criteria that provide the decision-informing advantage focus research on a metric, regardless of how well it represents its attribute.
—*Quality-Focused*. Criteria that provide the quality-focused advantage indicate that a metric is useful for improving the peoples' lives through high quality software.
—*Theory-Building*. The theory-building advantage enables researchers or practitioners to make general statements about the nature of software and its development.
—*Consensus Contribution*. The consensus contribution advantage highlights criteria that add to the common understanding of a metric's underlying attribute. This advantage also helps researchers agree as to how they define the underlying attribute, which in turn helps define the metric better. Does everybody understand how to collect and interpret the metric in the same way?

Table VI. Mapping from Criteria to Advantages

#	Criterion	Mathematical Soundness	Practicality	Correctness	Efficiency	Hypothesis-Strengthening	Meaningfulness	Decision-Informing	Quality-Focused	Theory-Building	Consensus Contribution	Difference-Detecting
1	A Priori Validity					X						
2	Actionability		X					X				
3	Appropriate Continuity	X					X				X	
4	Appropriate Granularity											X
5	Association							X		X		
6	Attribute Validity						X				X	
7	Causal Model Validity		X					X		X		
8	Causal Relationship Validity		X					X		X		
9	Content Validity						X				X	
10	Construct Validity			X							X	
11	Constructiveness							X		X		
12	Definition Validity										X	
13	Discriminative Power		X					X		X		
14	Dimensional Consistency	X					X					
15	Economic Productivity		X					X				
16	Empirical Validity					X				X		
17	External Validity							X	X	X		
18	Factor Independence					X	X					
19	Improvement Validity				X							
20	Instrument Validity	X		X								
21	Increasing Growth Validity	X									X	
22	Interaction Sensitivity	X									X	
23	Internal Consistency						X					
24	Internal Validity						X					
25	Monotonicity	X										
26	Metric Reliability			X								
27	Non-collinearity					X			X			
28	Non-exploitability		X									
29	Non-uniformity	X					X					X
30	Notation Validity			X							X	
31	Permutation Validity						X					
32	Predictability		X						X	X		
33	Prediction System Validity		X					X	X			
34	Product or Process Relevance		X									
35	Protocol Validity			X							X	
36	Rank Consistency					X		X	X			
37	Renaming Insensitivity						X					
38	Repeatability					X				X		
39	Representation Condition	X					X					
40	Scale Validity	X										

(Continued)

Table VI. (Continued)

#	Criterion	Mathematical Soundness	Practicality	Correctness	Efficiency	Hypothesis-Strengthening	Meaningfulness	Decision-Informing	Quality-Focused	Theory-Building	Consensus Contribution	Difference-Detecting
41	Stability			X								X
42	Theoretical Validity						X					
43	Trackability					X		X				
44	Transformation Invariance											X
45	Underlying Theory Validity									X		
46	Unit Validity						X					
47	Usability		X		X				X			

7.2. Criteria Application Process

To choose the appropriate metrics validation criteria for a specific metric, a researcher can perform the following six steps sequentially. Note that this methodology does not guarantee that a metric is valid for its intended use. Instead, this process acts as a guide for using the current state of the literature to demonstrate the advantages and limitations of a metric. Ultimately, a researcher who intends to validate a metric must consult the literature for a thorough analysis of that metric's validity.

(1) *Determine the intended use of the metric*. The researcher validating the metric examines how the metric ought to be used. Metrics' intended uses are numerous and vary according to specific practical or scientific contexts.
(2) Using Table VI, *highlight the advantages* that are appropriate for the intended use chosen in Step 1. Researchers can decide on the advantages that they wish to present to the research community or to a software development team.
(3) *Look up* the validation criteria that are tied to the advantages using Table VI.
(4) Carefully *choose validation criteria* from the list provided by Table VI while considering the intended use, relationships, and motivations in the community among the applicable criteria found in Sections 5 and 6. Do not choose criteria based on how well a metric performs, but rather only by the appropriateness of the criterion to the intended use.
(5) *Demonstrate* that the metric does or does not follow the chosen validation criteria.
(6) In presenting the metric in an academic publication, use the vocabulary of Table VI and Section 4 to *describe the advantages and limitations* of the metric in the context of the intended use identified in Step 1.

For a detailed example, suppose a researcher is proposing the metric `YearsExperience`: `YearsExperience` is equal to the sum of the number of years of software development experience on all of the members of the team. Following the steps just outlined, the process is as follows.

(1) The researcher determines the *intended use* of `YearsExperience` in evaluating the team's ability to develop high-quality software.
(2) In proposing this metric to the community, the researcher *highlights the advantage* that `YearsExperience` will save time and money when applied to a process (i.e., the Efficiency advantage described before).

(3) The researcher *looks up* the Efficiency advantage and discovers that it maps to Improvement Validity (#19) and Usabilty (#47) in Table VI.
(4) Both validity criteria apply to the intended use. Thus, to highlight the efficiency of `YearsExperience`, the researcher would *choose these two validation criteria*.
(5) The researcher then (a) examines other team metrics and *demonstrates* that `YearsExperience` is an improvement over current metrics (#19) and (b) *demonstrates* that `YearsExperience` can be applied to a software development process (#47).
(6) In presenting the metric's validation either to the team or to the academic community, the researcher then uses the vocabulary of Table VI (i.e., indicate that it is more efficient) to describe the metrics' *advantages and limitations*. Incidentally, the Usability criterion also has the advantage of Practicality and being Quality-Focused, two points that can also be argued in the researcher's work.

8. CONCLUSION

Our systematic literature review revealed that the academic literature contains a highly diverse set of assertions on what constitutes a valid metric. Our review resulted in 47 distinct validation criteria in the published software engineering literature, coming from 20 papers and 20 distinct authors. Some of the authors explicitly disagree with some criteria, while other criteria subsume each other. Researchers proposing new metrics should consider the applicability of the validation criteria in terms of our categorization and analysis. Understanding the opposing motivations behind the criteria helps us take the next steps toward establishing a generally-accepted standard set of validation criteria. Establishing a set of agreed-upon could bring metrics validation research from ad hoc analysis to a mature science of measurement and software improvement. However, after analyzing the criteria and the discussion among the authors of the papers, we conclude that metrics validation criteria provide answers to questions that researchers have about the merits and limitations of a metric.

REFERENCES

AYEWAH, N., HOVEMEYER, D., MORGENTHALER, J. D., PENIX, J., AND PUGH, W. 2008. Using Static Analysis to Find Bugs. *IEEE Softw. 25*, 5, 22–29.

BAKER, A. L., BIEMAN, J. M., FENTON, N., GUSTAFSON, D., MELTON, A., AND WHITTY, R. 1990. A philosophy for software measurement. *J. Syst. Softw. 12,* 3, 277–281.

BASILI, V. R. AND WEISS, M. 1994. A methodology for collecting valid software engineering data. *IEEE Trans. Softw. Engin. SE-IO, 6*, 728–738.

BRERETON, P., KITCHENHAM, B. A., BUDGEN, D., TURNER, M., AND KHALIL, M. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw. 80*, 571–583.

BRIAND, L., EMAM, K. E., AND MORASCA, S. 1996. On the Application of Measurement Theory in Software Engineering. *Empirical Softw. Engin. 1,* 1, 61–88.

BRIAND, L., EMAM, K. E., AND MORASCA, S. 1995. Theoretical and empirical validation of software product measures. Tech. rep. #ISERN-95-04. International Software Engineering Research Network.

BUSH, M. E. AND FENTON, N. E. 1990. Software measurement: A conceptual framework. *J. Syst. Softw. 12*, 3, 223–231.

CAVANO, J. AND MCCALL, J. 1978. A framework for the measurement of software quality. In *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*.

COURTNEY, R. E. AND GUSTAFSON, D. A. 1993. Shotgun correlations in software measures. *Softw. Engin. J. 8*, 1, 5–13.

CURTIS, B. 1980. Measurement and experimentation in software engineering. *Proc. IEEE 68*, 9, 1144–1157.

EL EMAM, K. 2000. A methodology for validating software product metrics. Tech. rep. NCR/ERC-1076, National Research Council of Canada, Ottawa, Ontario, Canada.

ELBAUM, S. G. AND MUNSON, J. C. 1998. Getting a handle on the fault injection process: validation of measurement tools. In *Proceedings of the 5th International Software Metrics Symposium*.

FENTON, N. 1994. Software measurement: a necessary scientific basis. *IEEE Trans. Softw. Engin. 20*, 3, 199–206.

FENTON, N. AND KITCHENHAM, B. 1991. Validating software measures. *J. Softw. Testing Verif. Reliab. 1,* 2, 27–42.

FENTON, N. AND NEIL, M. 1999. Software metrics: successes, failures and new directions. *J. Syst. Softw. 47*, 2–3, 149–157.

FENTON, N. AND NEIL, M. 2000. Software metrics: roadmap. Future of Software Engineering. In *Proceedings of the Conference on The Future of Software Engineering*.

HARMAN, M. AND CLARK, J. 2004. Metrics are fitness functions too. In *Proceedings of the 10th International Symposium on Software Metrics*.

HENDERSON-SELLERS, B. 1996. The mathematical validity of software metrics. *ACM SIGSOFT Softw. Engin. Notes 21*, 5, 89–94.

IEEE 1990. IEEE Standard 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.

ISO/IEC 1991. ISO/IEC 9126: Information technology - software product evaluation - quality characteristics and guidelines for their use.

JONES, C. 1994. Software metrics: good, bad and missing. *Computer 27*, 9, 98–100.

KITCHENHAM, B. 2004. Procedures for performing systematic reviews. Joint Tech. rep., Keele University Tech. rep. TR/SE-0401 and NICTA Tech. rep. 0400011T.

KITCHENHAM, B. 2010. What's up with software metrics? A preliminary mapping study. *J. Syst. Softw. 83*, 1, 37–51.

KITCHENHAM, B., PFLEEGER, S. L., AND FENTON, N. 1995. Towards a framework for software measurement validation. *IEEE Trans. Softw. Engin. 21*, 12, 929–944.

LINCKE, R. AND LOWE, W. 2006. Foundations for defining software metrics. In *Proceedings of the 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM'06)*.

MCCABE, T. 1976. A complexity measure. *IEEE Trans. Softw. Engin. SE-2*, 4, 308–320.

RAO, P. V. 2007. *Statistical Research Methods in Life Sciences*. Duxbury Press, Pacific Grove, CA.

ROCHE, J. M. 1994. Software metrics and measurement principles. *ACM SIGSOFT Softw. Engin. Notes 19*, 1, 77–85.

SCHNEIDEWIND, N. F. 1991. Validating software metrics: producing quality discriminators. In *Proceedings of the International Symposium on Software Reliability Engineering.*

SCHNEIDEWIND, N. F. 1992. Methodology for validating software metrics. *IEEE Trans. Softw. Engin. 18*, 5, 410–422.

WEYUKER, E. J. 1988. Evaluating software complexity measures. *IEEE Trans. Softw. Engin. 14*, 9, 1357–1365.

Received September 2010; revised May 2011; accepted June 2011