Software Metrics Validation Criteria: A Systematic Literature Review

Andrew Meneely
Ben Smith
Laurie Williams

North Carolina State University
(apmeneel, bhsmith3, lawilli3)@ncsu.edu

Abstract

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 paper 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 categorized each validation criterion into three categories and nine sub-categories. We present an analysis of the 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. We have determined in this paper that, rather than arbitrarily choosing validation criteria for each metric, researchers should choose criteria that confirm that the metric is appropriate for its intended use. Researchers proposing new metrics should consider the applicability of the validation criteria in terms of our categorization and analysis.

Keywords: software metrics, validation criterion, systematic literature review

1. Introduction

Practitioners and researchers alike use software metrics to both improve and understand software products and the software development process. 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 merits of a metric, the software engineering community will 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, Pfleeger et al. 1995). *The objective of this paper 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 focusing on 20 publications that propose or indicate software metrics validation criteria. Our review is a useful guide for two audiences: 1) *metrics researchers*: software engineering researchers who propose, use, and validate new metrics; and 2) *metric validation researchers*: software engineering researchers seeking to propose methodologies, criteria, or frameworks for validating metrics.

Metrics researchers will want to use this review as:

- **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 engineering 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 thinking about metrics.

Software *metrics validation researchers* will want to use this review as:

- **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 summarizing the validation criteria, we have summarized the discussion surrounding metric validation. We found that this discussion has not concluded and wish to re-invigorate the discussion amongst the top minds in our field 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. Certain holes and weaknesses exist in the overall hierarchy, and new criteria could be proposed that address these holes and weaknesses.

This paper 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. In Section 4, we present the sources from our review. We present the extracted criteria in Section 5 and the mapping of their relationships in Section 6. Next, we present the common themes and conflicts we discovered in Section 7. Finally, we conclude by outlining the opposing philosophies we have identified in the literature in Sections 8 and 9.

2. Terms and Definitions

During our review, we found many different usages and definitions for the same words. We define our usage of the following words here in Section 2.1. Next, we define examples that we refer to throughout the paper 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 to the product itself exhibits, for example, the size of code (ISO/IEC 1991) (Fenton 1991).

- **External attribute:** Attributes that are dependent on the behavior of the system, for example, the reliability of a system 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 (Fenton 1991) points out that metric has been used as:
  - a number derived from a product, process or resource;
  - a scale of measurement
  - an identifiable *attribute* (see above).

  Also, a metric measures the degree to which a system, component or process possesses a given *attribute* (see above) (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 "co-relating" 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). However, the results of this review indicate that the community does not agree on a universal definition of metric validation. We include the ISO/IEC definition for clarity and general understanding. Metric validation, as we view it, is the ensuring that a metric is acceptable by the community in both interpretation and use because it is well-grounded, relevant, meaningful and logically correct.

2.2 Examples

In our discussions throughout this paper, we use several 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 (N. Ayewah, D Hovemeyer 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 recommends a procedure for conducting systematic literature reviews (Kitchenham 2004), 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). Additionally, we consider the following three critical aspects of the literature review: *foundation in the literature, abstraction, and backwards informative.* Figure 1 illustrates the conceptual structure of our review as a pyramid, with the tiers of the pyramid representing the steps we performed in the study.

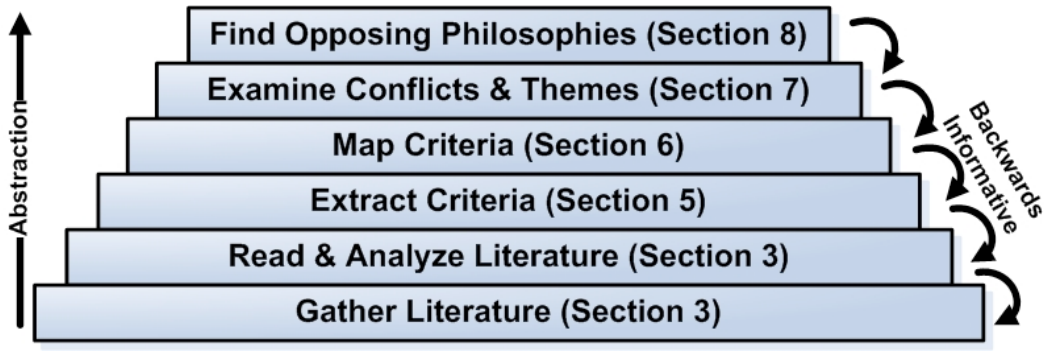

Figure 1: Building a review with foundation in literature

The structure of our review in Figure 1 has the following characteristics:

- **Foundation in literature:** Our foundation is in the concrete evidence of proposed metrics validation criteria and surrounding discussion; we achieved these conceptual ideas through synthesis and analysis of these sources available in the literature. We have intentionally not projected our own ideas onto what previous researchers have said.
- **Abstraction:** The review began at the concrete stage of gathering papers. As the review proceeded, we traveled into increasingly higher layers of abstraction until reaching the conceptual stage of deriving opposing philosophies.

- **Backwards Informative:** Occasionally, a more abstract process would give us new insight into a more concrete process after its completion, and we would revisit these ideas to help solidify the foundation. For example, once we had settled on a set of sources to be the basis of our review, we decided to consider the sources that were cited by our original set in the review process.

The following sub-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

Based upon Kitchenham's systematic review process (Kitchenham 2004), the first phase of planning a systematic literature review is to identify the research objective that should be satisfied by conducting the review. The objective of our review was to provide a practical summary of the metrics validation criteria found in the academic 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. (Brereton, Kitchenham et al. 2007), we used the following search engines to conduct our review:

Google Scholar (http://scholar.google.com) CiteSeerX (http://citeseerx.ist.psu.edu/)

IEEExplore (http://ieeexplore.ieee.org) 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.
4. Otherwise, continue to the next set of 10 results (step 3).

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.

We created the form shown in Figure 2 to assess the quality of our primary sources and identify sources that fit our research objective.

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? (See Desired vs. Necessary Criteria in Section 7.4)* • *Does this paper exhibit scientific objectiveness, 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

Figure 2: Primary Source Full-Text Review Form.

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 1. The total results column in Table 1 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 2,288 sources. We accepted only 536 of the 2,288 sources we saw in the search of step one.

Table 1: 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**

Several types of sources came up after completing the first pass of the initial search. Of the 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. CiteSeerX does not indicate the type of source in its results, and these sources did not warrant further collection of bibliographic information, as they were all eliminated from the review in a later phase. The next few phases of the search are summarized by Figure 3.

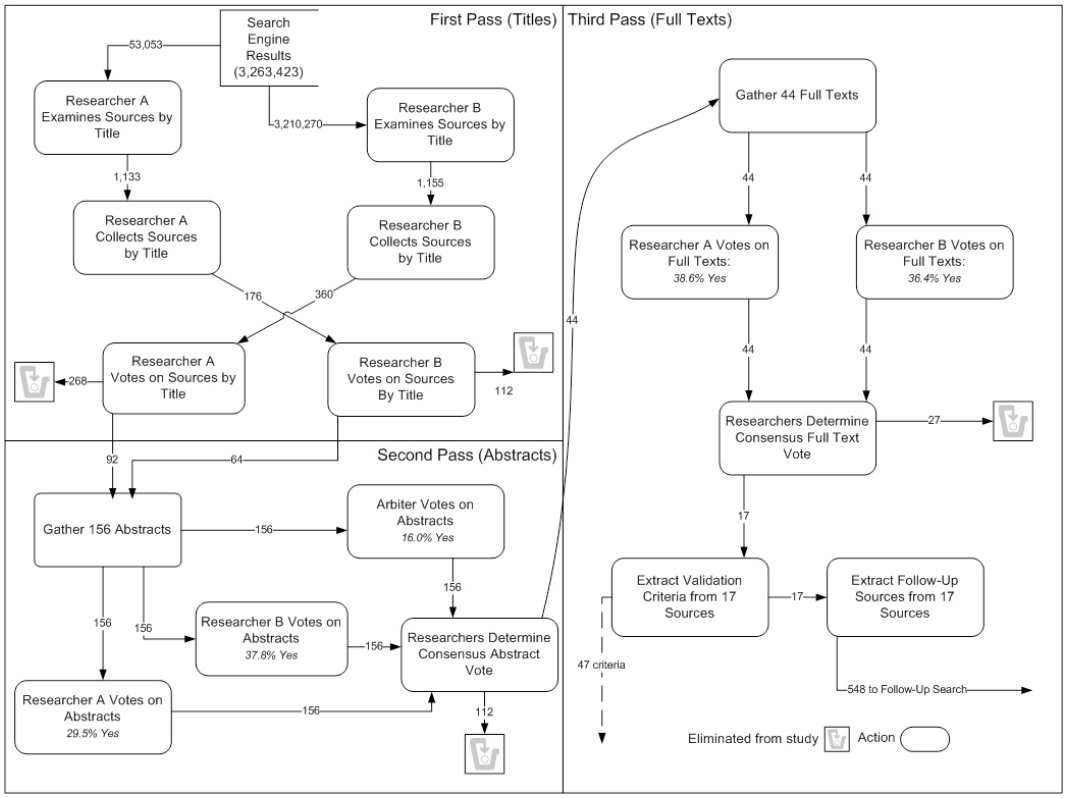

Figure 3: Overall view of the source-gathering procedure used in initial literature review

Next, we voted to confirm the 536 titles for relevance. As prescribed by Brereton, et al. (Brereton, Kitchenham et al. 2007), Researcher A voted on whether the results found by Researcher B were relevant to the review, and vice versa. Researcher A collected 176 unique titles, of which Researcher B accepted 64 (thus eliminating 112). Researcher B collected 360 sources, of which Researcher A accepted 92 (thus eliminating 268). Thus, we accepted a total of 156 titles from the voting confirmation to proceed to the next phase of the review.

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 2. Researcher A voted yes to 46 of 156 abstracts and Researcher B voted yes to 59 of 156 abstracts. The researchers agreed on 102 abstracts and disagreed on 54. A meeting was held to form consensus on the 54 contested abstracts. We voted 40 abstracts as irrelevant and vote 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.

Table 2: Second pass voting results

Voter	Yes (Keep)	No (Discard)
Researcher A	46	110
Researcher B	59	97
Researchers A & B Agreed	102	54
Meeting Votes	14	40
Overall Consensus	51	105

The overall consensus on the abstracts, as determined above, was passed on to the Arbiter, the third author in this paper. As shown in Table 3, 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 17 abstracts and rejected 18 abstracts. We accepted 44 abstracts total (rejecting 112), and proceeded to the next phase of the review.

Table 3: Second pass voting with arbiter results

Voter	Yes (Keep)	No (Discard)
Researchers	51	105
Arbiter	38	118
Agreed	121	35
Meeting Votes	17	18
Overall Consensus	44	112

3.2.3 Third Pass: Full Texts.

The next phase of the search was to gather the full text for each of the 44 sources whose abstract the research team agreed upon as shown in Table 4. 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.

Table 4: 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

In summary, we searched a population of over 3 million sources, examined 2,288 titles, and selected 536 sources. 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 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.

4.0 Sources and Metadata

Our review revealed 20 sources regarding metrics validation criteria. An empirical analysis of the years of the publications (see Figure 4) reveals that a large number of writings were conducted on the topic of software engineering metrics validation criteria between the years of 1987-1995, and since that time the publicized research on metrics validation has declined.

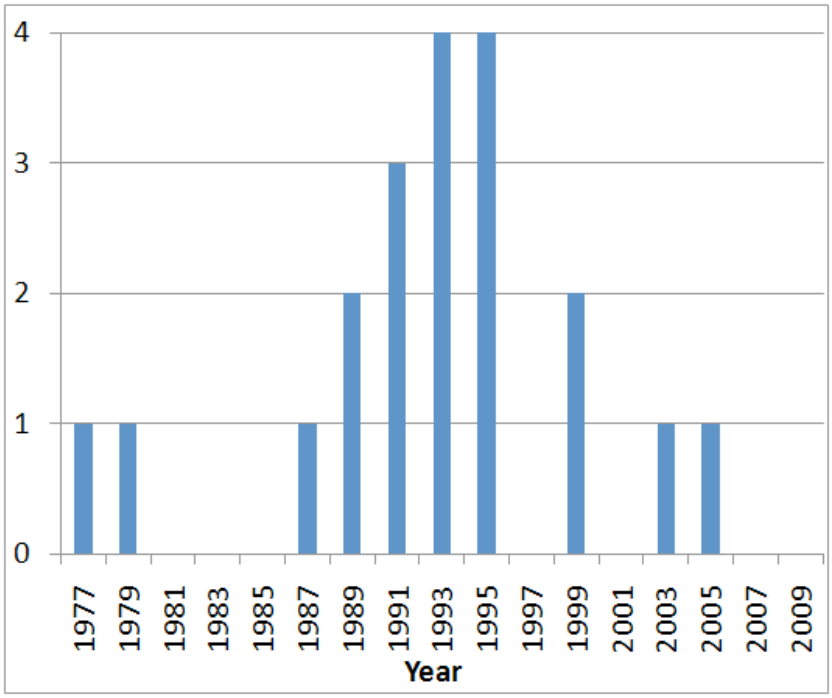

Figure 4: Number of sources over time

Furthermore, the papers involved significant cross-referencing, discussion, and disagreement, which we present visually using a citation network (see Figure 5), which is a directed graph of our sources. Each source is represented by a node, where the name is comprised of the first five letters of the first author's name along with the year. A node is connected to another node if the two nodes cite each other. Upon visual inspection, the majority of the sources cite a paper published a year or two prior.

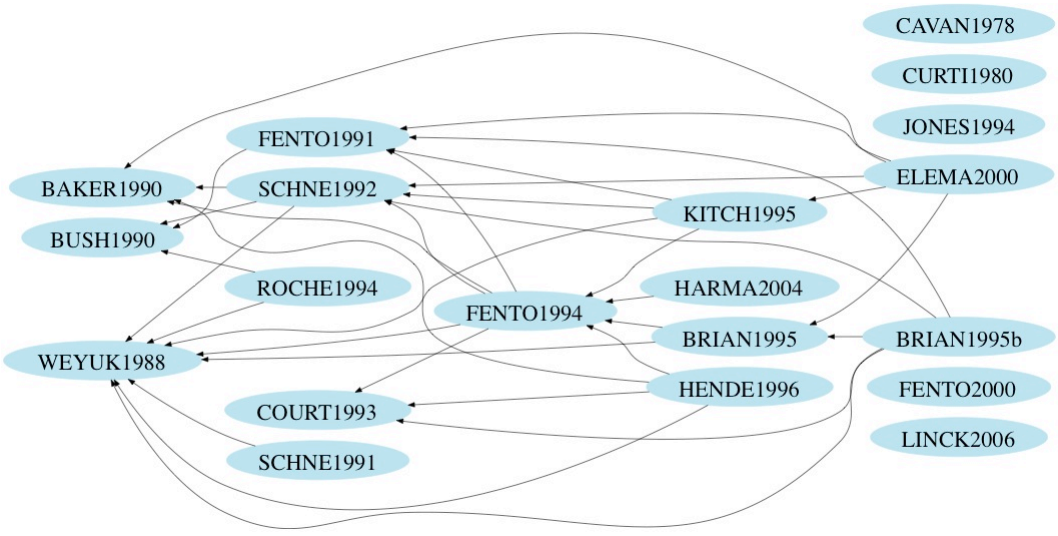

Figure 5: Citation network for the 20 sources in our study.

Additionally, we provide the number of Google Scholar "cited by" values[1] for the 20 sources in Figure 6. These "cited by" values offer a rough approximation of how often the sources in our survey have been cited since their publication. The median number of citations for the sources in our study was 54.5, and the standard deviation was 132.4.

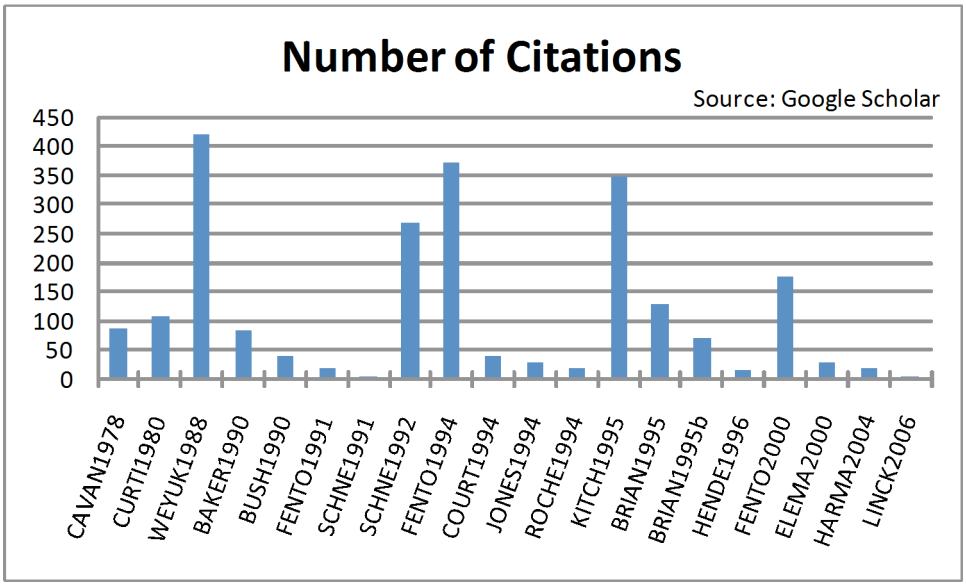

Figure 6: Number of Google Scholar citations, sorted chronologically

Also, we find that insight can be gained by looking at the tag cloud for the raw text of the 20 sources, as shown in Figure 7. A tag cloud[2] is a graphical representation of words in which the size of the word is relative to the frequency of occurrence in the text. The larger a word, the more frequently it appears in the text. The text for the source of this cloud was not altered in any way, so page numbers, references, and copyright information are included in the cloud.

[1] Citation counts gathered on 9/24/2009

[2] Generated by <http://tagcrowd.com/>

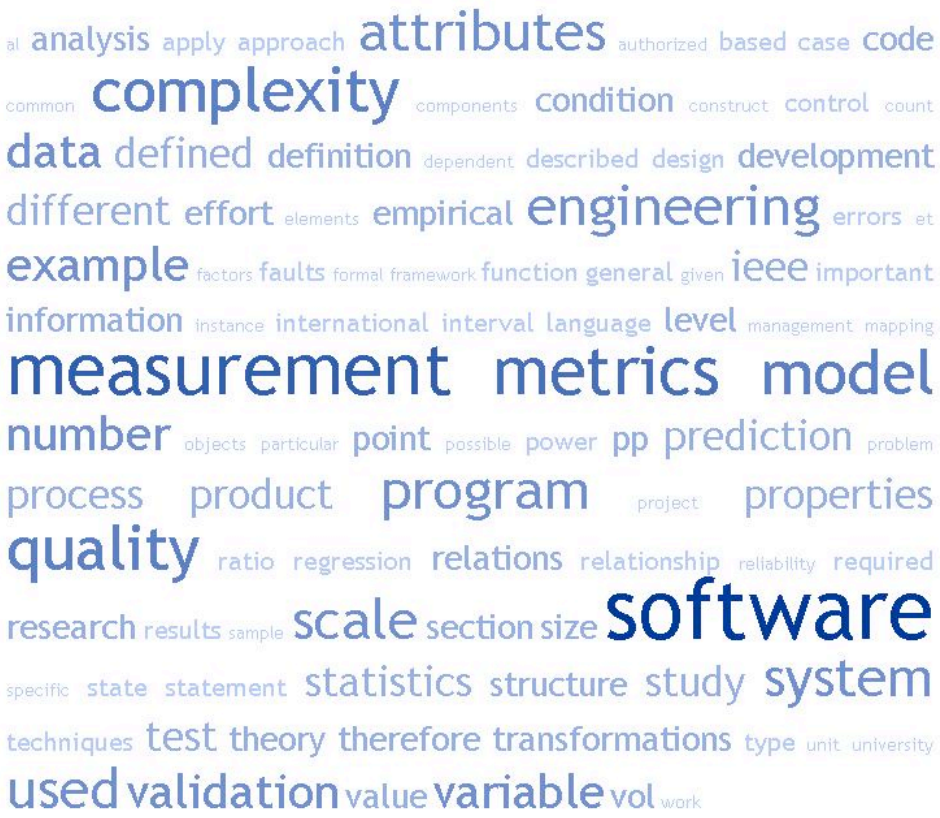

Figure 7: A tag cloud generated from the full text of the 20 sources in review

This tag cloud helps demonstrate some salient points about the sources included in our study:

1. The words "software", "measure", "metrics", and "model" are the four largest words, indicating that the concatenation all the text of our review is on point with the objective of our research.
2. The word "complexity" is inordinately large, which corroborates an observation we made anecdotally: many authors writing about metrics validation are also writing about complexity metrics. A deeper discussion of code complexity can be found in Section 7.2.
3. The word “quality” appears prominently, implying that many of the papers included a discussion of a metrics being related to software quality.
4. The word “validation” is not as large as one might expect. However, since the sources we included in our review are writing *about* validation, and specifically what it comprises, the papers usually do not need to repeatedly mention it by name.

5. Extracting the Validation Criteria

We read 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. If two criteria were closely related but discussed slightly different concepts, we left them separate.

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 10 to act as a) a reference point for this paper and for future research; and b) an invitation 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 10 in alphabetical order. We further synthesize the resultant criteria in Section 6.

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

Figure 10: List of all 47 validation criteria found in the review

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. For the rest of this paper, a reference to a number with a pound sign denotes a reference to one of these criteria (e.g. #30 refers to predictability).

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, Emam et al. 1995), (Baker, Bieman 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 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 word "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, Pfleeger 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, Pfleeger et al. 1995), (Weyuker 1988). This criterion is a grouping of three 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 a given 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), (Schneidewind 1992), (Fenton 1994). We use the term "direct" to imply that the metric is measured without the use of a model. 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, Pfleeger et al. 1995), (Baker, Bieman 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 pre-determined critical value* (Schneidewind 1991), (Schneidewind 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, Pfleeger 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 a) the intended measurement of a metric; or b) 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, Emam et al. 1995), (Baker, Bieman 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 6.1.

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, Pfleeger 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 2000) (Baker, Bieman 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 6.1.

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. **Non-collinearity**: *A metric has **non-collinearity** 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. **Non-exploitability**: *A metric exhibits **non-exploitability** 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 non-exploitability, which is not to be confused with our use of stability (#39).

29. **Non-uniformity**: *A metric has **non-uniformity** 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, Bieman et al. 1990), (Fenton 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, Emam et al. 1995), (Schneidewind 1991), (Schneidewind 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, Pfleeger et al. 1995). The example that Kitchenham et al. provides 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), (Schneidewind 1992), (El Emam 2000) Schneidewind defines the repeatabilty 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, Pfleeger et al. 1995), (Fenton 1994), (Bush and Fenton 1990), (Harman and Clark 2004), (Baker, Bieman 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 (Fenton 1994) describes the Representation Condition as a two-way correspondence between a metric and an attribute. Kitchenham (Kitchenham, Pfleeger et al. 1995) states 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, Emam et al. 1995), (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 Farenheit 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, Pfleeger et al. 1995), (Briand, Emam et al. 1995), (Briand, Emam et al. 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 6.1.

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, Pfleeger et al. 1995), (Baker, Bieman et al. 1990), (Roche 1994). 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, Pfleeger et al. 1995), (Fenton 1994). For example, fault

rate may be used to measure program correctness or test case effectiveness (Kitchenham, Pfleeger 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.

6. Mapping the Validation Criteria

The spectrum of metrics validation criteria presented in Section 5 indicate 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. 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 11. 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. A more detailed presentation of this map including a tracing to the original sources can be found in Appendix A.

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, in for any given category, a criterion may be presented in the future. For example, (shown in Figure 11) 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, two criteria required multiple categorizations, including empirical (#16) and theoretical (#42), which are described in detail in the following section.

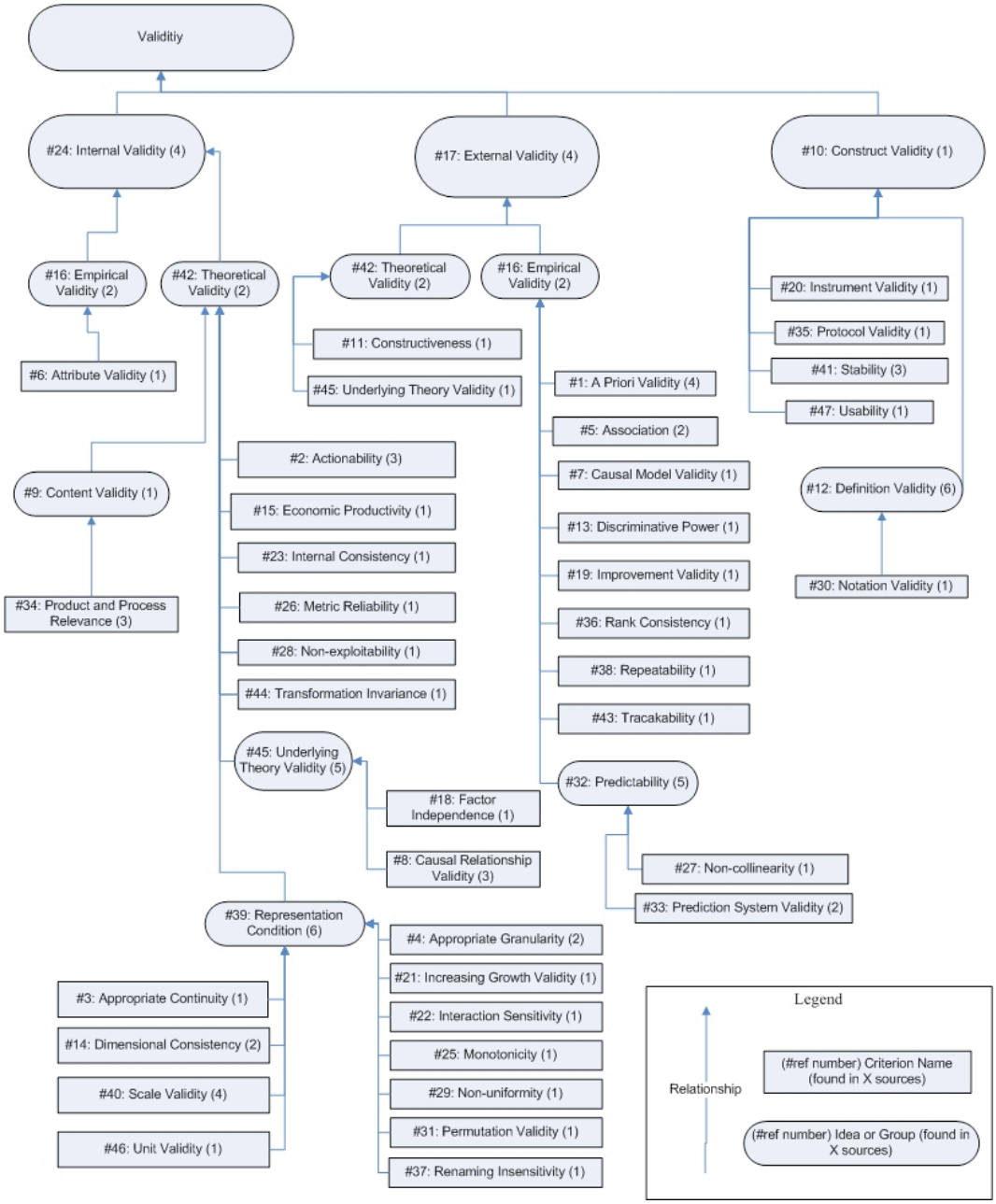

Figure 11: The 47 criteria, categorized with number of references in parentheses

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

6.1 Internal/External vs. Theoretical/Empirical

Beneath our top-level categories are THEORETICAL validity (#42), and empirical validity (#16). While many of the 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 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.

6.2 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 a poor instrument (#20).

Also in the construct validity category is definition validity (#12), which deals with providing a clear, unambiguous, deterministically-defined metric. Many authors (Lincke and Lowe 2006), (Henderson-Sellers 1996), (Baker, Bieman 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. Among the reasons for definition validity is that 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.

6.3 Representation Condition and Internal, Theoretical Validity

Within internal validity, theoretical validity is the Representation Condition. The Representation Condition 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, Bieman et al. 1990) claim that satisfying the Representation Condition is the *same* as internal, theoretical validity. However, in our review we found several criteria that are not concerned directly with Representation Condition, but could still be considered 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.

Kitchenham, et al. also explain that one way of corroborating a measure (perhaps satisfying the Representation Condition in the process) is to perform experiments to see whether people agree that an attribute exists or whether mapping to a value captures their understanding of the attribute (#6) (Kitchenham, Pfleeger et al. 1995). Finding disagreements in such an experiment would underscore the importance of theoretical validity as some authors define metrics for concepts such as size, complexity, cohesion, coupling, but do so without providing a precise definition for the *attribute* they intend to measure. Without precise definitions, software measurement becomes a matter of belief. If two people disagree on the concept of, say, code size, then they will certainly disagree on a metric. For example, Weyuker (Weyuker 1988) highlights the difficulty in evaluating complexity metrics because not everybody agrees on what a complexity metric is truly supposed to be measuring. A more thorough discussion of complexity is provided in Section 7.2.

6.4 External, Empirical 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, Bieman 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).

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.

A priori validity (#1) and underlying theory (#45) are also related ideas in the context of external, empirical validity. The difference is that the a priori validity criterion states that the underlying theory must be presented *before* performing a correlation, and the underlying theory criterion merely states that a correlation must *have* an underlying theory.

While Schneidewind's trackability validation criterion (#43), which was not mentioned often by other sources in our study, bears resemblance to the causality criterion (#8). The trackability validation criterion 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). Although a metric can be trackable and still lack causality, one can use the contrapositive of the causality implying trackability to argue that a metric *lacks* causality (i.e. not trackable implies not causal). For example, one could invalidate the claim that low test coverage causes faults by showing that fixing a fault in code does not change the test coverage. The fact that test coverage trackability is invalid causes test coverage trackability to be causally invalid.

7. Conflicts and Common Themes

As we proceeded to analyze the criteria from our study, we noticed that several ideas were recurrent in our sources. In some instances, such as with code complexity, the sources have differing views about a particular validation criterion or group of criterion. In this section, we outline these cases by comparing and contrasting the perspectives we observed.

7.1 Measurement Theory

A common line of reasoning in many of the sources in our study involves a discussion of measurement theory. Measurement theory is a field taken from mathematics that deals with developing sound measurements. Within measurement theory, there are different scales that a metric can take the form of, which provides a set of "admissible transformations" for that scale. Five major scale types are often discussed: (Briand, Emam et al. 1995), (Fenton 1994):

- **Nominal**: is a classification of the objects without the notion of order or distance between the classes. No transformations of the given nominal metric are admissible other than equality. Schneidewind's discriminative power (#13) is based on this measurement principle (Schneidewind 1992).
- **Ordinal**: is a ranking of the classified objects. Equality and inequality (e.g. <, >) are admissible transformations. Schneidewind's ranking consistency (#36) is based on this measurement principle (Schneidewind 1992).
- **Interval**: a metric with ordinal scale type, with the distances between the objects being meaningful, but the measures themselves are not. Admissible transformations include equality, inequality, addition, and subtraction. The Prediction (#32) criterion uses the interval scale to predict a factor value and the ratio scale for measuring prediction accuracy. Association (#5) also uses the interval scale for measuring the difference in component quality (Schneidewind 1992).
- **Ratio**: a metric with an interval scale and a meaningful "zero" value, so that ratios between values are meaningful. Admissible transformations include equality, inequality, addition, subtraction, multiplication, and division.
- **Absolute:** a metric has an exact mapping to the attribute it is measuring; all transformations preserve meaning.

Curtis (Curtis 1980) explains that statistical techniques make no assumptions about the type of scale employed, so the problem of scale type validity is one in measurement theory rather than statistical theory. Curtis's statement supports what Kitchenham, et al. (Kitchenham, Pfleeger et al. 1995), Fenton, et al. (Fenton 1994) and other authors indicate regarding scale validity. Kitchenham, et al. also suggest that accuracy is an important criterion for one scale type over another. Furthermore, Kitchenham, et al. recommend that measures should be subject to some measurement error determined by the accuracy of the chosen scale type (Kitchenham, Pfleeger et al. 1995). Many other authors discuss measurement theory and scale types (El Emam 2000) (Bush and Fenton 1990) (Fenton 1991) (Briand, Emam et al. 1995) (Schneidewind 1992).

7.2 The Complexity Controversy

Among all of the sources we reviewed, the most frequently cited work was Weyuker's (Weyuker 1988) complexity metrics validation criteria (#4a-c, 21, 22, 25, 29, 31, 37). Although Weyuker's nine criteria are specifically defined over code complexity measures, we include Weyuker's criteria in this paper for three reasons:

1. Weyuker is the only complexity-based author found in our study who proposed criteria for validation of complexity metrics;
2. Her criteria have been intensely contested by other metrics validation researchers over the years; and

3. Her criteria are closely related to the Representation Condition, a property that many researchers cite as the most important category in internal metrics validation.

Kitchenham, et al. (Kitchenham, Pfleeger et al. 1995) provided a thorough examination of the complexity metrics validation criteria, generalizing some criteria to all metrics while also rejecting several criteria. Kitchenham, et al. contend that Weyuker's Monotonicity (#25) and Interaction Sensitivity (#22) are inadmissible since they imply a specific scale type (Kitchenham, Pfleeger et al. 1995). Kitchenham, et al. also regard Increasing Growth (#21) as inadmissible because it involves the relation "less than" (i.e. "<") and thereby excludes nominal scale units (Kitchenham, Pfleeger et al. 1995). Kitchenham, et al. explain that Renaming Insensitive (#37) is related to the Representation Condition and that accepting the Representation Condition means Renaming Insensitive (#37) is unnecessary. Kitchenham, et. al also explain that Weyuker's Permutation (#31) is incorrect because she is confusing program correctness with structural complexity (Kitchenham, Pfleeger et al. 1995).

The issue of the necessary scale types in code complexity is a much-discussed topic. Briand, et al. (Briand, Emam et al. 1995) support Weyuker's complexity metric validation criteria by indicating that complexity metrics need not be *additive* (i.e. on an interval scale). Note that they did not to indicate that metrics **cannot** be additive, just that additivity should not be a mandatory requirement for complexity metrics. The statement by Briand, et al. is in direct contradiction with the argument presented by Fenton (Fenton 1994) which indicates that complexity metrics must be additive.

Furthermore, transformation invariance (#43) is a specific instance of Weyuker's Renaming Insensitivity (#36). For example, renaming a variable preserves semantics of a program. However, transformation invariance can include other semantics-preserving transformations, which Harman and Clark provide an example of. Furthermore, Weyuker's appropriate granularity criteria (#4) are meant to capture the notion that different implementations of the same functionality can have different measurements, which is a different concept than *semantically* equivalent programs having the same metric. As a result, transformation invariance can be considered a generalization of renaming insensitivity, but bounded by appropriate granularity.

Fenton (Fenton 1994) describes several weaknesses in Weyuker's approach, claiming that it attempts to characterize several incompatible views of complexity. Fenton also explains that Weyuker's granularity criteria (#4) are from "incompatible views of complexity" (Fenton 1994). Fenton also explains that Weyuker's axiomatic approach to complexity is prescribing necessary but not sufficient criteria for the validation of complexity metrics.

7.3 "External Validation is All We Need"

Another contested part of the literature is the metrics validation framework presented by Shneidewind (Schneidewind 1991) (Schneidewind 1992) (#5, #13, #32, #36, #38, #43). Most of the disagreements with Schneidewind's framework are not in the necessity of his six validation criteria, but in the initial claim that they are sufficient. That is, the claim that external (#17), empirical (#16) validation is enough to consider a metric valid. Central to the disagreement is the

following statement: “If metrics are to be of greatest utility, the validation should be performed in terms of the quality functions (quality assessment, control, and prediction) that the metrics are to support” (Schneidewind 1992).

Specifically, Kitchenham, et al. (Kitchenham, Pfleeger et al. 1995) explain that Schneidewind's method of validating software metrics is incomplete as validating a predictive model is different from validating a measure. However, Kitchenham, et al (Kitchenham, Pfleeger et al. 1995) agree that basic statistical techniques such as correlation analysis can be used to investigate relationships between attributes. Fenton (Fenton and Neil 2000) disagrees with Schneidewind by stating that valid metrics may be inherently poor at predicting software quality.

None of the authors in our sources, however, seem to disagree with the correctness of any of Schneidewind's six validation criteria, rather, they extend his framework. Briand, et al. (Briand, Emam et al. 1995) explain that once a measure is valid from a theoretical point of view, the measures must be validated empirically as no theoretical model can guarantee truthfulness, supporting Schneidewind’s emphasis on external, empirical validation. Lastly, El Emam (El Emam 2000) presents a more detailed version of Schneidewind's (Schneidewind 1991), (Schneidewind 1992) approach by demonstrating various techniques to find statistical correlation. Roche (Roche 1994) agrees with Schneidewind (Schneidewind 1992) when he says "associated with the validation process is on-going validation"; that is, a metric should be validated continuously on various projects and through time, which Schneidewind supports in describing repeatability (#38).

7.4 Necessary, Desirable, and Sufficient Criteria

In extracting and mapping the validation criteria, we noticed that some authors presented their criteria with varying levels of necessity. Some of the criteria are presented as a ways to invalidate a metric rather than describing what is *sufficient* for validation. Furthermore, some authors presented their criteria as *necessary* for validation, while others treated their criteria as *desirable*. In this paper, we examine three relationships between a validation criterion and metric validity: “sufficient”, “necessary”, and “desirable”.

- A **sufficient** validation criterion S is defined by extending the formal logic definition of "sufficient" (Barwise 2002): if a metric M satisfies S, then M is valid. However, if M *does not* satisfy S, it is not necessarily invalid. Informally, satisfying a sufficient criterion is enough consider a metric valid, but it may not be the only criterion that exists.
- A **necessary** validation criterion N is defined by extending the formal logic definition of "necessary" (Barwise 2002): if a metric M *does not* satisfy N, then M is invalid. However, if M satisfies N, we cannot say that M is (fully) valid. Informally, a metric must satisfy a necessary criterion to be considered valid, but satisfying a necessary criterion is not enough to consider the metric valid.

- A **desirable** validation criterion D is defined by extending the definition of (Weyuker 1988): if a metric M *does not* satisfy D, then M is not necessarily invalid, however, metric M satisfying D enables M to be interpreted or used in a specific way.

For example, Kitchenham et al. (Kitchenham, Pfleeger et al. 1995) regard the Representation Condition (#39) as necessary, but Cavano (Cavano and McCall 1978) presents constructiveness (#11) as desirable.

Furthermore, even when multiple authors discussed the same criterion, some would treat the criterion as necessary; others would treat the criterion as desirable. For example, (Kitchenham, Pfleeger et al. 1995) refers to the criteria in (Weyuker 1988) as being necessary, while Weyuker et al. (Weyuker 1988) refer to their criteria as desirable.

We also noticed that the authors attempted to find conditions that are desirable and necessary, but nobody presents sufficient criteria. While Schneidewind presents his criteria as necessary (Schneidewind 1991), (Schneidewind 1992), Briand, et al. have argued that he implies his criteria are sufficient as well (Briand, Emam et al. 1995). Indeed, Briand, et al. have indicated that there should be no all-encompassing set of criteria which is to be deemed sufficient when they write "Measurement concepts that are different in nature should be characterized by different sets of properties" Briand, et al. (Briand, Emam et al. 1995).

Ultimately, not every author explicitly stated how crucial their validation criterion was, so we could not assemble a full classification of "desired versus necessary" criteria without inferring ideas that the authors never conveyed. However, the observation warranted a deeper analysis of why the authors felt that certain criteria were more important than others. This deeper analysis brought us to collect our authors' motivations into a spectrum of philosophies, which we discuss in the following section.

8. 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 for 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.

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 (Fenton 1994) and Briand et al. (Briand, Emam 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 (Schneidewind 1991) and Roche (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; 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 adequately effective results.

- **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.

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 (Briand, Emam et al. 1995).
- **Metrics should be generally applicable**. A metric must be universally defined before we can begin to apply it. For example, Lincke (Lincke and Lowe 2006) contends 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 12.

Figure 12: The spectrum of diametrically-opposed philosophies

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 paper. 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), (Schneidewind 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 sub-criteria 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.

9. Choosing Validation Criteria: A Scenario

Suppose a researcher, John, proposes a new metric named SuperM. In his analysis, John wishes to demonstrate to the audience that the metric can be used to prevent security vulnerabilities from being injected into the software project. In his first study, John introduces the metric with an underlying theory of how the metric is related to post-release vulnerabilities (A priori validity, #1). John shows a statistical association (Association, #5) between the metric and post-release vulnerabilities. This association indicates that the metric is worth investigating further, so John a prediction system (Prediction system validity, #33) to show that SuperM empirically predicts post-release vulnerabilities with an acceptable amount accuracy.

Next, John introduces SuperM to a development team, to demonstrate its usefulness in an industrial setting. The developers are asked to follow SuperM and use it to guide their manual

inspections. On the first day of the study, John notices that developers can "fool" SuperM by quickly copy-and-pasting dead code to inflate the metric's values (Non-exploitability, #28). To achieve non-exploitability and thereby protect the truthfulness of the metric's values, the developer creates SuperM2.

A separate researcher, Jane, upon examining our list of metrics validation criteria, comes across the repeatability criterion (Repeatability, #38) and decides to test the predictability and association for SuperM2 with several software projects. Jane's study shows that SuperM2 is repeatable across all but one of the projects in the study. This result shows that while SuperM2 does extend beyond the context of the initial project, it may not be applicable to all projects.

Meiyappan presents a study on SuperM2 and shows that, in the Java programming language, an infinite number of programs could have the same SuperM2 value. As a result, Meiyappan recommends not using SuperM2 in a Java context. Furthermore, the Meiyappan notices that some implementations of SuperM2 produce negative values, but SuperM2 does not make sense for negative values. Meiyappan investigates John's work and, upon confirming that SuperM2 is not defined in the negative range (Appropriate Continuity, #3), submits a patch to the original tool to fix the instrument (Instrument Validity, #20).

Each one of the studies in this scenario was driven by one factor: a pre-determined set of validation criteria from this literature review. In each of the steps of the progression above, a researcher initially had a question about the metric he or she wished to have answered. Next, the researcher decided on a set of metric validation criteria that would answer the question he or she had about the metric. After evaluating the metric using the chosen metrics validation criteria, the researcher was able to provide evidence for the answer to the question he or she initially posed. The end result is an evolutionary process of articulating the merits and limitations of a proposed metric.

10. 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. Understanding the opposing motivations behind the criteria helps us take the next steps toward establishing software metric validation criteria. Researchers proposing new metrics should consider the applicability of the validation criteria in terms of our categorization and analysis. Establishing a set of necessary and/or sufficient criteria 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.

10. Sources

- (Kitchenham, Pfleeger et al. 1995) B. Kitchenham, S. L. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation," IEEE Transactions on Software Engineering, vol. 21, pp. 929-944, 1995.
- (Curtis 1980). B. Curtis, "Measurement and experimentation in Software Engineering," Proceedings of the IEEE, vol. 68, pp. 1144-1157, 1980.
- (Roche 1994) J. M. Roche, "Software metrics and measurement principles," ACM SIGSOFT Software Engineering Notes, vol. 19, pp. 77-85, 1994.
- (Weyuker 1988) E. J. Weyuker, "Evaluating software complexity measures," IEEE Transactions on Software Engineering, vol. 14, pp. 1357-1365, 1988.
- (Jones 1994) C. Jones, "Software metrics: good, bad and missing," Computer, vol. 27, pp. 98-100, 1994.
- (Fenton 1994) N. Fenton, "Software measurement: a necessary scientific basis," IEEE Transactions on Software Engineering, vol. 20, pp. 199-206, 1994.
- (Schneidewind 1991) N. F. Schneidewind, "Validating software metrics: producing quality discriminators," in International Symposium on Software Reliability Engineering, 1991, pp. 225-232.
- (El Emam 2000) K. El Emam, "A Methodology for Validating Software Product Metrics," National Research Council of Canada, Ottawa, Ontario, Canada NCR/ERC-1076, June, 2000.
- (Fenton and Neil 2000) N. E. Fenton and M. Neil, "Software metrics: roadmap," in Future of Software Engineering, Limerick, Ireland, 2000, pp. 357-370.
- (Briand, Emam et al. 1995) L. Briand, K. E. Emam, and S. Morasca, "On the application of measurement theory in software engineering," International Software Engineering Research Network Tech Report #ISERN-95-04, 1995.
- (Schneidewind 1992) N. F. Schneidewind, "Methodology for validating software metrics," IEEE Transactions on Software Engineering, vol. 18, pp. 410-422, 1992.
- (Henderson-Sellers 1996) B. Henderson-Sellers, "The mathematical validity of software metrics," ACM SIGSOFT Software Engineering Notes, vol. 21, pp. 89-94, 1996.
- {Briand, 1995 #230} L. Briand, K. E. Emam, and S. Morasca, "Theoretical and empirical validation of software product measures," International Software Engineering Research Network, Technical Report, 1995.

- (Harman and Clark 2004) M. Harman and J. Clark, "Metrics are fitness functions too," in 10th International Symposium on Software Metrics, 2004, pp. 58-69.
- (Cavano and McCall 1978) J. Cavano and J. McCall, "A framework for the measurement of software quality," in Software quality assurance workshop on functional and performance issues, 1978, pp. 133-139.
- (Lincke and Lowe 2006) R. Lincke and W. Lowe, "Foundations for defining software metrics," in 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM '06), Genoa, Italy, 2006.
- (Baker, Bieman et al. 1990) A. L. Baker, J. M. Bieman, N. Fenton, D. Gustafson, A. Melton, and R. Whitty, "A philosophy for software measurement," Journal of Systems and Software, vol. 12, pp. 277-281, 1990.
- (Courtney and Gustafson 1993) R. E. Courtney and D. A. Gustafson, "Shotgun Correlations in Software Measures," Software Engineering Journal, vol. 8, pp. 5-13, 1993.
- (Bush and Fenton 1990) M. E. Bush and N. E. Fenton, "Software measurement: A conceptual framework," Journal of Systems and Software, vol. 12, pp. 223-231, 1990.
- (Fenton 1991) N. Fenton, "Validating software measures," Journal of Software Testing, Verification & Reliability, vol. 1, pp. 27-42, 1991.

11. References

Baker, A. L., J. M. Bieman, et al. (1990). "A philosophy for software measurement." Journal of Systems and Software **12**(3): 277-281.

Barwise, J., Etchemendy, J (2002). Language, Proof, and Logic, Center for the Study of Language and Information.

Basili, V. R. and Weiss (1994). "A methodology for collecting valid software engineering data." IEEE Transactions on Software Engineering **SE-IO**(6): 728-738.

Brereton, P., B. A. Kitchenham, et al. (2007). "Lessons from Applying the Systematic Literature Review Process Within the Software Engineering Domain." Journal of Systems and Software **80**: 571-583.

Briand, L., K. E. Emam, et al. (1995). On the application of measurement theory in software engineering, International Software Engineering Research Network Tech Report #ISERN-95-04.

Briand, L., K. E. Emam, et al. (1995). Theoretical and empirical validation of software product measures, International Software Engineering Research Network, Technical Report.

Bush, M. E. and N. E. Fenton (1990). "Software measurement: A conceptual framework." Journal of Systems and Software **12**(3): 223-231.

Cavano, J. and J. McCall (1978). A framework for the measurement of software quality. Software quality assurance workshop on functional and performance issues.

Courtney, R. E. and D. A. Gustafson (1993). "Shotgun Correlations in Software Measures." Software Engineering Journal **8**(1): 5-13.

Curtis, B. (1980). "Measurement and experimentation in Software Engineering." Proceedings of the IEEE **68**(9): 1144-1157.

El Emam, K. (2000). A Methodology for Validating Software Product Metrics, National Research Council of Canada, Ottawa, Ontario, Canada NCR/ERC-1076, June.

Elbaum, S. G. and J. C. Munson (1998). Getting a handle on the fault injection process: validation of measurement tools. Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International.

Fenton, N. (1991). "Validating software measures." Journal of Software Testing, Verification & Reliability **1**(2): 27-42.

Fenton, N. (1994). "Software measurement: a necessary scientific basis." IEEE Transactions on Software Engineering **20**(3): 199-206.

Fenton, N. E. and M. Neil (2000). Software metrics: roadmap. Future of Software Engineering, Limerick, Ireland.

Harman, M. and J. Clark (2004). Metrics are fitness functions too. 10th International Symposium on Software Metrics.

Henderson-Sellers, B. (1996). "The mathematical validity of software metrics." ACM SIGSOFT Software Engineering 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,." International Organization for Standardization and the International Electrotechnical Commission.

Jones, C. (1994). "Software metrics: good, bad and missing." Computer **27**(9): 98-100.

Kitchenham, B. (2004). "Procedures for Performing Systematic Reviews." Joint Technical Report, Keele University Technical Report TR/SE-0401 and NICTA Technical Report 0400011T.

Kitchenham, B., S. L. Pfleeger, et al. (1995). "Towards a Framework for Software Measurement Validation." IEEE Transactions on Software Engineering **21**(12): 929-944.

Lincke, R. and W. Lowe (2006). Foundations for defining software metrics. 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM '06), Genoa, Italy.

McCabe, T. (1976). "A complexity measure." IEEE Transactions on Software Engineering **SE-2**(4): 308-320.

N. Ayewah, D Hovemeyer, et al. (2008). "Using Static Analysis to Find Bugs." IEEE Software **25**(5): 22-29.

Roche, J. M. (1994). "Software metrics and measurement principles." ACM SIGSOFT Software Engineering Notes **19**(1): 77-85.

Schneidewind, N. F. (1991). Validating software metrics: producing quality discriminators. International Symposium on Software Reliability Engineering.

Schneidewind, N. F. (1992). "Methodology for validating software metrics." IEEE Transactions on Software Engineering **18**(5): 410-422.

Weyuker, E. J. (1988). "Evaluating software complexity measures." IEEE Transactions on Software Engineering **14**(9): 1357-1365.

Appendix A: Mapping Authors, Criteria and Categories

We present the full mapping of our criteria, from the sources in our study in Figures 13, 14, 15, and 16. An arrow between a source and a criterion or a group of criteria indicates that the source mentioned or purported the idea or criterion. An arrow between a criterion and an idea or group indicates that the criterion is a part of or specific instance of that group.

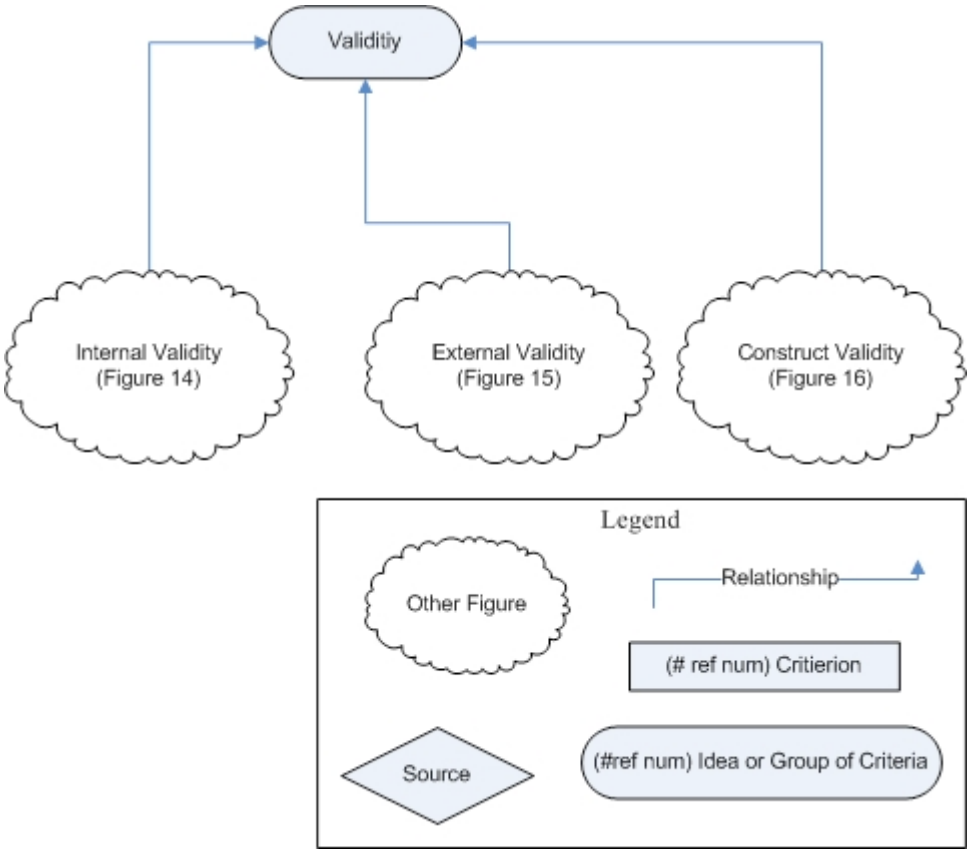

Figure 13: The metrics map at high level, with legend

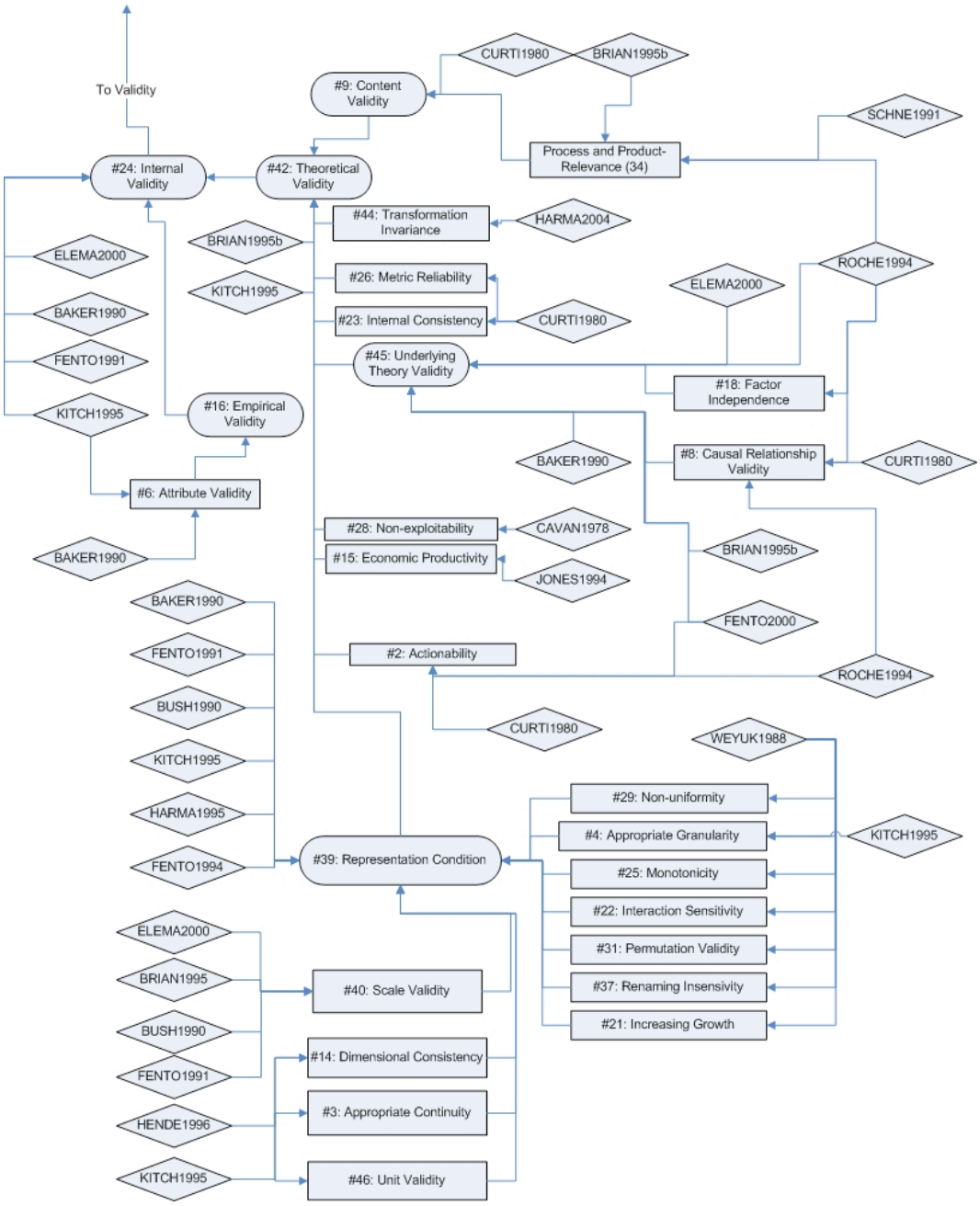

Figure 14: Internal metrics validation criteria, with authors included

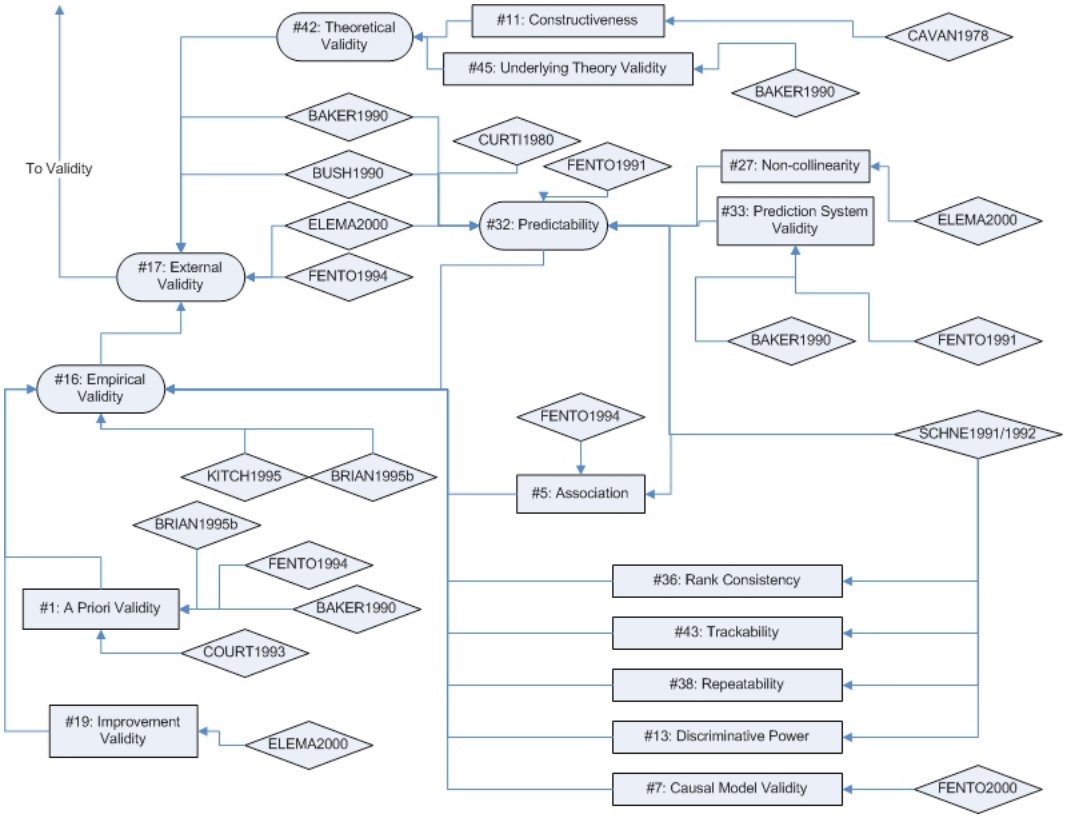

Figure 15: External metrics validation criteria, with authors included

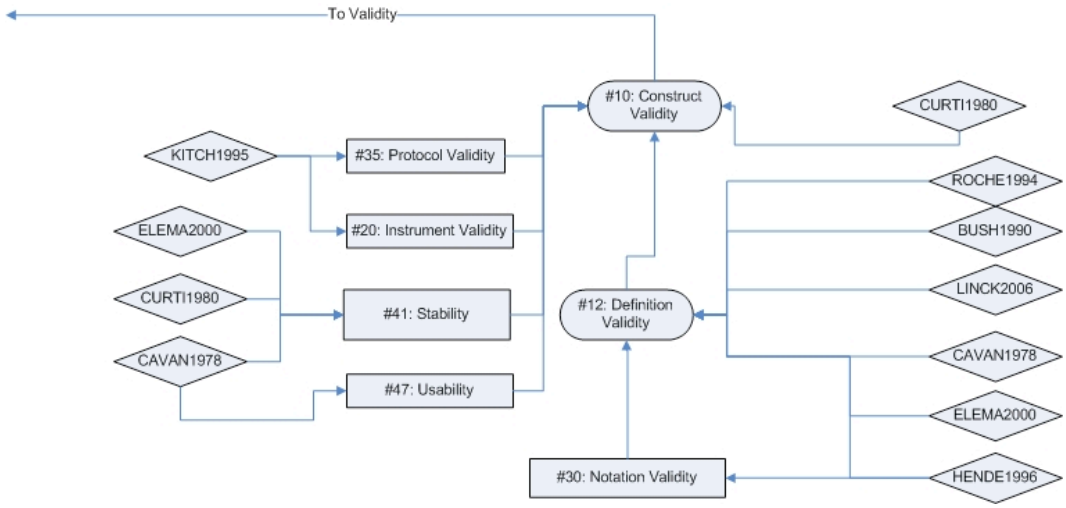

Figure 16: Construct validity criteria, with authors included