The Journal of Systems and Software 83 (2010) 37–51

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

What's up with software metrics? – A preliminary mapping study

Barbara Kitchenham

Dept. Computer Science and Mathematics, Keele University, Staffordshire ST05, UK

ARTICLE INFO

Article history:
Received 25 October 2008
Received in revised form 27 April 2009
Accepted 26 June 2009
Available online 1 July 2009

Keywords:
Software metrics
Secondary study
Literature survey
Mapping study
Influential papers
Empirical evaluation problems

ABSTRACT

Background: Many papers are published on the topic of software metrics but it is difficult to assess the current status of metrics research.
Aim: This paper aims to identify trends in influential software metrics papers and assess the possibility of using secondary studies to integrate research results.
Method: Search facilities in the SCOPUS tool were used to identify the most cited papers in the years 2000–2005 inclusive. Less cited papers were also selected from 2005. The selected papers were classified according factors such as to main topic, goal and type (empirical or theoretical or mixed). Papers classified as "Evaluation studies" were assessed to investigate the extent to which results could be synthesized.
Results: Compared with less cited papers, the most cited papers were more frequently journal papers, and empirical validation or data analysis studies. However, there were problems with some empirical validation studies. For example, they sometimes attempted to evaluate theoretically invalid metrics and fail to appreciate the importance of the context in which data are collected.
Conclusions: This paper, together with other similar papers, confirms that there is a large body of research related to software metrics. However, software metrics researchers may need to refine their empirical methodology before they can answer useful empirical questions.

1. Introduction

In 2004, my colleagues and I suggested that we needed to consider adopting evidence-based approach to software engineering (Kitchenham et al., 2004; Dybå et al., 2005). Although evidence-based practice arose first in the field of medicine, it seems natural for software engineering to make use of the increasing body of empirical software engineering research in order to become more evidence-based. Evidence-based software engineering (EBSE) relies on empirical software engineering research but emphasizes the need to *find and aggregate* best available evidence on a specific topic and uses secondary studies such as systematic literature reviews and mapping studies as the methodological framework for finding and aggregating evidence (Kitchenham and Charters, 2007).

The difference between systematic literature reviews and mapping (scoping) studies is the type of question they answer. A systematic literature review asks a fairly specific question such as "Which of the Chidamber and Kemerer OO metrics are good predictors of fault-proneness", while mapping studies ask more general questions such as "What do we know about OO related metrics". Systematic literature reviews and mapping studies are called secondary studies because they aim to aggregate the research of other studies which are referred to as primary studies. It must be emphasized that EBSE does not simply equate to the secondary study methodology. The aim is to help practitioners in industry. However, practitioners ask somewhat different, and usually very context specific, questions, for example "Will metric *X* be a good predictor of fault-proneness in my large evolving object-oriented system?" Moving from the results of a systematic literature review to answering a practitioner-oriented question is non-trivial.

This paper presents a preliminary mapping study of software metrics research. It aims to identify and categorise influential software metrics research and also to consider the opportunity for aggregating research papers. Thus it addresses two research questions:

1. What papers are currently most influential in the software metrics research community?
2. Can software metrics papers on a similar topic be meaningfully aggregated?

I refer to this as a preliminary study because I make no claim for completeness. I have concentrated on the years 2000–2005 and have used only the SCOPUS tool to search for relevant papers. SCOPUS is a very useful system, because it indexes IEEE, ACM, and Elsevier publications. For software engineering researchers this means it indexes many of the leading publications, including all those used by researchers such as Sjøberg et al. (2005) and Glass et al. (2004) who have undertaken major literature studies of software engineering research. It also indexes other important software engi-

E-mail address: barbara.kitchenham@cs.keele.ac.uk

0164-1212/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2009.06.041

neering sources of empirical studies such as the Empirical Software Engineering Journal and the Springer Lecture Notes on Computer Science series. This makes SCOPUS potentially a very powerful tool for research. Furthermore, researchers who act as reviewers for the Journal of Systems and Software and Information and Software Technology are rewarded by being given 30 days use of SCOPUS.

Since 2003, there have been four literature surveys in the metrics field that are described in Section 2. The main novelty of this paper is that it uses citation analysis to identify influential papers and investigates the opportunities for aggregating results by detailed review of a particular subset of papers addressing the evaluation of software metrics. I describe the methodology in Section 3, and the results of classifying the selected papers in Section 4. Section 5 provides a discussion of the results, concentrating on discussing the problems of attempting to aggregate the evaluation papers. Section 6 concludes the paper.

2. Related work

There have been four recent literature surveys of software metrics (Bellini et al., 2008; Catal and Diri, 2009; Gómez et al., 2008; Purao and Vaishnavi, 2003). All these studies are mapping studies categorising metrics papers in various dimensions, and identify frequencies and trends over time, but they vary substantially in rigour.

Purao and Vaishnavi (2003) concentrate on product metrics for object-oriented systems. Although they have surveyed the metrics literature, theirs is not a *systematic* mapping study because there is no defined search process. Their main concern is to develop a framework and notation to define metrics. It is difficult to determine how many primary studies contributed to their study but looking at the tables that link concepts to references they appear to have used 17 different primary studies.

They present a measurement framework to organise OO metrics, based on an internal perspective comprising *entities* and *attributes* and an external perspective comprising *questions* and *goals*. These are linked in two ways: a *Development Process* **has** *goals* **and** produces *entities* and a *Mapping Function* (i.e. Metric) provides a relationship between the internal and external perspectives. They provide a mathematic formalisation for discussing metrics that produces a generic name expressed as *Entity<as>State(s).Attribute/Metric* and a standard formalism for defining the calculation needed to construct metrics. Counts of the product metrics they discovered in the literature were tabulated against the entity/attribute scheme showing the relationships between the metrics and

- The stages of software development.
- The metric type (direct or indirect, elementary or composite).

In addition, the metrics definition notation was used to investigate.

- Aggregation relationships between metrics.
- The use of one metric to compute another.

This is a very theoretical paper that aims at a complete description of all possible metrics but does consider whether the defined measures have any inherent value to software engineers or managers.

Bellini et al. (2008) perform a more systematic search process, but did not explicitly discuss the results of their search and have a narrative style aggregation process that does not show a clear link between individual papers and their conclusions. They do not confirm how many primary studies contribute to their study but have a total of 122 references.

Their discussion is informed by the SWEBOK (Abran et al., 2004) and Fenton and Pfleeger's application of measurement theory to software measurement (Fenton and Pfleeger, 1997). They first discuss the role of software measurement within software engineering as an aid to management planning, prediction and control. They use Fenton and Pfleeger's classification of metrics as relating to product, process or resource, and having either an internal or an external focus. They present a set of information that should be used to fully define any measure within an overall discussion of the Goal-Question-Metric (GQM) approach and discuss the MEDEA extension to GQM method (Briand et al., 2002). They finally discuss how to collect and analyse measures, in particular they criticise over-reliance on scales of measurement to guide analysis. They conclude that measurement needs to take a broader view not concentrating on technical issues but, following the lead of the Information Systems field, also considering user, task, organisational and environment factors.

Gómez et al. (2008) undertook a more standard mapping study with a defined search process, inclusion criteria, and a standard data extraction process. They included 78 primary studies and summarise the metrics found in these papers in the following dimensions:

- Entity type: product (79%), process (21%), project (29%).
- Attributes measured: most frequent being complexity (19%), size (16%), inheritance (8%), defect (7%), structuredness (7%), time (5%), others less than 4%.
- Validation methods: empirical (46%), theoretical (26%), theoretical and empirical (28%).
- Measurement focus: OO (33%), process (16%), quality (16%), structure (15%), project (11%), complexity (3%), function points (2%), UML (1%), OCL (1%),
- Lifecycle phase: analysis (5%), design (4%), development (27%), testing (12%), maintenance (14%).
- Project managers book of knowledge phases: initial (48%), intermediate (36%), final (16%).

The most disappointing aspect of the paper is that the 78 primary studies are not explicitly cited. This may be because the paper is a conference paper and would have had restrictions on number of pages.

Catal and Diri (2009) provide a more focussed mapping study looking only at software fault prediction studies. They have implemented a well-organised systematic mapping review with well-defined research questions. They identify the sources searched, their inclusion criteria and their classification categories and reference all the 27 journal papers and 47 conference papers they included in their literature review. The only methodological weakness is that their search process is not well-defined and the classification information is not shown for each paper. They report distributions before and after 2005, since that was the year that the PROMISE repository (which publishes software engineering data sets) was established.

They report the following results:

- The journals that published more than two fault model papers are: IEEE transaction of software engineering (9); software quality journal (4); journal of systems and software (3); empirical software engineering (3)
- 14% of papers were published before 2000 and 86% after.
- Types of data sets used by authors were: private (60%), partial (8%), public (31%), unknown (1%). "Partial" means data from open source projects that have not been circulated. Since 2005 the proportion of private datasets has reduced to 31%, the proportion of public data sets has increased to 52%. There are 14% partial datasets and 3% unknown.

- Data analysis methods are machine learning (59%), statistics (22%), statistics and machine learning (18%) and statistics and expert opinion (1%). After 2005 the distribution of methods is machine learning (66%), statistics (14%), statistics and machine learning (17%) and statistics and expert opinion (3%).
- 60% of papers used method level metrics, 24% used class level metrics, 10% were file level metrics, other categories less than 5%. 2005, 53% were method level, 24% were class level and 17% were file level (others less than 3%).

They conclude that:

- More studies should use class-level metrics to support early prediction (which assumes that class metrics are valid early predictors).
- Fault studies should use public datasets to ensure results can be repeatable and verifiable. This seems a reasonable requirement.
- Researchers should increase usage of machine learning techniques. This would seem to require that machine learning models are in some sense better than statistical models but this is not investigated in the study.

Overall, the results in this paper should have most similarity with Gómez et al.'s study (Gómez et al., 2008). However, Gómez et al. base their counts on metrics, whereas the study unit herein is the primary study itself.

3. Research method mapped

This mapping study was based on citation analysis and classification of relevant papers. Trend analysis was based on comparing frequently cited and infrequently cited papers.

3.1. Identifying relevant papers

I used SCOPUS to search for software metrics papers published in 2005 and the most cited software metrics papers in the years 2000–2005. The search processes started on 16th September 2008 and ended on 26th September 2008. The search process is outlined in Table 1.

3.1.1. Papers published in 2005

Search 1 found 505 articles many of which were clearly irrelevant – for example many papers reported computer systems that collect or analyze biological data or hardware data, such as Shaneck et al. paper on wireless sensors (Shaneck et al., 2005). A random selection of 50 papers found 10 that were relevant to software metrics suggesting somewhere in the region of 100 relevant papers in the 505 selected.

After using SCOPUS to sort the papers in terms of relevance, to be on the safe side, I screened the first 200 papers finding 128 papers that looked relevant based on the title and a brief review of the abstract. A more detailed review of the abstract and keywords found 87 papers and, after the test–retest validation described in Section 3.1.3, resulted in selecting 91 relevant papers.

It was fairly difficult to determine objective inclusion and inclusion criteria. The initial inclusion criteria were:

- Related directly to the collection (including extraction and storage), development (including specification), evaluation or analysis of measurements derived from software abstractions (requirements specifications, architectures, design, code).
- Related to the use of software metrics for other software engineering purposes (e.g. re-engineering or fault prediction)
- Tools or frameworks for the specification, extraction, storage, analysis and/or use of software metrics.

The initial exclusion criteria were:

- Quality of Service metrics (since these are primarily related to the performance and reliability of deployed systems).
- Papers concerned with performance optimization of deployed systems.

These criteria, however, were unable to cope with papers related to defect prediction, or reliability that used process metrics such as number of changes or faults during development. I decided to include such papers although some researchers may regard software metrics as referring specifically to metrics derived from software representations. As a result of this decision, I also included other papers related to software process metrics including the process maturity scales.

The classification system for papers was developed using the original set of 87 papers using a mixed top-down and bottom-up approach. Starting with some initial ideas about types of metrics papers and then reviewing the abstracts and keywords of each paper, I developed the classification scheme discussed in Section 3.2.

3.1.2. Most frequently cited papers

For the years 2000–2005 (inclusive), I sorted the selected papers on the basis of the citation count and manually reviewed the titles (and if necessary the abstracts) of the papers selecting the most frequently cited software metrics papers.

There was no pre-determined number of papers to be selected in each year. I selected between 10 and 18 papers ensuring that when I reached a cut-off point I selected all relevant papers with the same number of citations. Obviously, restricting the number of papers to about 15 meant that the cut-off level decreased as the years increased, as shown in Table 2.

3.1.3. Selection validation

Search 1 and Search 2F were performed independently. This allowed me to undertake a test–retest check of the papers comparing the most cited papers found by Search 1 with the most cited papers found by Search 2F (see Table 3).

Ignoring the paper that was not available for selection in Search 1, a Kappa analysis found that a disagreement in 3 out of 13 papers was unacceptable (both expected agreement and actual agreement was 76.9%, giving a Kappa value of 0).

Therefore I re-did all the manual screening for most cited papers, and found four additional papers in 2001, one additional pa-

Table 1
Search process.

Search Id	Goal	SEARCH string	Purpose
Search 1	Metrics papers 2005	(TITLE-ABS-KEY(software) AND TITLE-ABS-KEY(metrics OR measurement)) AND PUBYEAR IS 2005 AND SUBJAREA(COMP)	Comparison of least and most cited papers for classification and trend analysis Assessment of proposed classification scheme Trialling inclusion/exclusion criteria
Search 2A–2F	Metrics papers 2000–2005	TITLE-ABS-KEY(software) AND TITLE-ABS-KEY(metrics OR measurement)) AND PUBYEAR IS *X* AND SUBJAREA(COMP) where *X* identifies the specific year	Identification of most cited papers for classification and trend analysis

Table 2
Cut-off point for citations and number of papers selected per year.

Year	Cut-off number of citations	Number of papers selected	Revised number of papers selected
2000	19	16	16
2001	15	16	20
2002	12	15	19
2003	10	14	17
2004	9	12	15
2005	6	12	16

Table 3
Test-retest results.

	Search 1	Search 2F
Relevant most cited papers found	14	12
Missing papers	1	2
Extra papers found after test–retest	1	2

per in 2002, three additional papers in 2004 and four additional papers in 2005. The final number of frequently cited papers is shown in the final column of Table 2.

3.2. Data extraction

I collected some standard information about all papers:

- The authors.
- The full reference.
- Whether the paper was a conference or journal paper.
- The total number of number of citations.
- For the most cited papers, I also recorded the number of citations excluding self citations.

The number of citations changes quite rapidly for some papers. However, when I collected data about the additional most-cited papers (found after the test–retest check), I did not update all the citation records for the other papers.

Other data recorded for each paper were classification data (see the next section).

3.2.1. Classifying the papers

After looking at the abstracts and keywords of papers published in 2005, I classified the papers according to the following criteria:

- The main topic (see Table 4).
- Whether the paper was empirical, theoretical or both (see Table 5).
- Whether the paper was about object-oriented metrics.
- Whether the paper applied metrics in the context of open source software.
- For analysis papers, whether the techniques discussed were statistical or AI based and the specific technique(s) being used.
- Whether the context of the paper was Maintenance (which I took to include system evolution). Note this was collected for the 2005 papers after I extracted data for the most cited papers and noticed a possible trends towards maintenance related papers.

For the most cited papers I also recorded:

- What sort of metrics were being developed/evaluated. This turned out not to be very helpful (with the exception of development and sometimes evaluation papers) since many papers considered a wide variety of metrics or were relevant to any metric.

Table 4
Main topic of metrics papers.

Categories	Category meanings
Development	The paper is about the specification of a new (or revised) metric or set of metrics
Evaluation	The paper is about the evaluation, validation or assessment of existing metrics
Analysis	The papers discuses and/or illustrates methods of analysing software metrics data either to suggest new or improved methods, or to critique existing methods
Framework	The paper is about general procedure or process by which sets of metrics are defined, extracted, stored and/or analysed
Tool	The paper is about an automated framework
Program	The paper is about establishing, improving, and/or evaluating a measurement program
Use	The paper is about processes that use metrics for some other software engineering purpose
Literature Survey	The paper summarises the literature on some aspect of metrics

Table 5
General type of paper.

Categories	Meaning
Empirical	The paper describes the results of analysing software metrics data. Typically papers that evaluate an existing metrics or set of metrics are included in this category
Theoretical	The paper is descriptive. It discusses some issue and may (but not always) consider some theoretical issues concerning software metrics. It does not include an empirical study of the issue being discussed. Typically tools and framework papers are included in this category but also evaluation papers that use measurement theory
Both	The paper is a mixed theoretical and empirical paper. Typically papers that development metrics and provide some evaluation or demonstration of the metric are included in this category

- The general goal of the study. In this case, I did not attempt to construct a classification framework, I just tried to summarise the papers in a reasonably consistent way. This lead initially to the identification of some general categories (see Table 6), but many papers had very specific goals (e.g. the development of a specific type of metric) and did not fit any of the general categories.

3.2.2. Data extraction validation

I extracted data from all the relevant papers identified by Search 1 (irrespective of citation count) before extracting data from the most cited papers. After extracting data from most cited papers in the years 2000–2004, I extracted data from the most cited papers in 2005. This allowed me to do a test–retest check on some of the extracted data. There were no discrepancies between the data related to the type of paper (Journal or Conference) and whether the papers concerned OO metrics, or open source systems.

Table 6
Goals.

Categories	Meaning
Fault prediction	Papers that attempt to predict the number of defects in components
Component fault classification	Papers that attempt to identify components as fault-prone or not
Sizing papers	Papers concerned with measures of program size
Effort estimation	Papers concerned with predicting effort for new projects or maintenance activities
Re-engineering	Papers that are concerned with the use of metrics to improve products. This includes papers about re-factoring, and autonomic computing

Table 7
Kappa analysis for two classifications.

Statistics	Main topic	General type
Agreement	69.3%	76.8%
Expected Agreement	21.03%	31.9%
Kappa	0.67	0.66
Z (significance)	4.47 ($p < 0.0001$)	3.78 ($p < 0.001$)

However, there were some discrepancies between the classification of main topic and general type as shown in Table 7.

The statistics indicate that the agreement was acceptable; nonetheless, after revising the classification used for 2005 top cited papers, I checked the main topic for all the most cited papers and reclassified two papers as analysis papers rather than evaluation papers since they were concerned with the modeling methods used to undertake metrics evaluations rather than evaluating metrics.

3.3. Aggregating primary study results

Although mapping studies tend to stop at categorizing identified papers, the goal of systematic literature review is to attempt to aggregate results into a coherent body of knowledge that allows researchers and practitioners to answer specific metrics-related questions. A goal of this article is to assess whether such aggregation is possible in the case of software metrics. The initial searches identified analysis and evaluation papers as the type of papers which are of most interest to the metrics community. The analysis papers include some of my own publications, which might lead to a lack of objectivity. So I studied the evaluation papers in more detail by reading each one and extracting a standard set of information from each paper with the aim of assessing the extent to which the results could be integrated.

4. Results

In this Section 1 present some tabulations of the results of categorising the identified papers and present a more detailed study of papers classified as evaluation studies.

4.1. Most cited papers

The most cited papers for years 2000–2005 are shown in citation order in Tables 23–28 in Supplementary material. I cross-tabulated these papers to investigate:

- The relationship between paper type and citations (see Table 8).
- The relationship between main topic and citations (see Table 9).
- The relationship between main topic and paper type (see Table 10).
- The publication source for conference and journal papers (see Tables 11 and 12). The journals that published most of the papers overlap with the journals identified by Catal and Diri (2009). The main difference was that Catal et al. included Software Quality Journal in their top four journals whereas (Table 12) includes IEEE software.

Table 8
Citations and paper type for the most cited papers.

Number of citations	Number of Journal papers	Number of Conference papers	Total
<10	10	2	12
10–19	31	11	41
20–29	17	4	21
30–39	8	2	10
40–49	10	0	10
50+	6	0	6
Total	81	19	100

Table 9
Citations against main topic for most cited papers.

Topic	Number of citations						Total
	<10	10–19	20–29	30–39	40–49	50+	
Analysis	4	10	10	3	0	0	27
Development	3	5	3	2	1	1	15
Evaluation	2	10	2	1	5	4	24
Framework	0	4	2	1	1	0	8
Programs	0	2	0	0	0	0	2
Literature survey	0	1	0	0	0	0	1
Tool	1	2	0	0	1	0	4
Use	2	7	4	3	2	1	19
Total	12	41	21	10	10	6	100

Table 10
Main topics against paper type for most cited papers.

Topic	Paper type			Total
	Empirical	Theoretical	Both	
Analysis	15	5	7	27
Development	0	5	10	15
Evaluation	16	2	6	24
Framework	0	6	2	8
Programs	1	0	1	2
Literature survey	0	1	0	1
Tool	0	2	2	4
Use	14	1	4	19
Total	46	22	32	100

Table 11
Source of conference papers for most cited papers.

Conference (or source)	Number of papers
International Software Engineering Conference	8
Conference on Software Maintenance	3
Proceedings of European Conference on Software Maintenance and Re-engineering	3
Others (1 paper only)	5
Number of different sources	8

Table 12
Sources of journal papers for most cited papers.

Journal	Number of papers
IEEE TSE	34
Journal of Systems and Software	9
Empirical Software Engineering	8
IEEE Software	5
Information and Software Technology	4
Computer	2
ACM Transactions on Software Engineering Methodology	2
IEE Proceedings Software	2
IEEE Trans on Parallel and Distributed Systems	2
IEEE Trans on Reliability	2
Others (1 paper only)	11
Number of different sources	20

These tables show that the most cited papers:

- Were usually journal papers (81%).
- Were most frequently analysis papers (27%) or evaluation papers (24%).

- Were mostly empirical papers (46%) or papers that involved theoretical and empirical results (32%). Only 22% were entirely theoretical.
- That IEEE transactions on software engineering included by far the largest number of influential journal papers (42%) compared with the next highest ranking journal (JSS) which published 11% of the influential journal papers.

Evaluation papers were among the most frequently quoted papers (i.e. 9 out of the 16 papers that were cited more than 40 times). In addition, 67% of evaluation studies were empirically based, 25% were both empirical and theoretical, whereas only 8.3% (i.e. 2 papers) were entirely theoretical. This suggests more emphasis on empirical validation compared with Gómez et al. who found 26% of metrics were only evaluated theoretically.

The analysis papers showed a completely different distribution of methods to those reported by Catal and Diri (2009). Twenty of the 27 (i.e. 75%) papers classified as "analysis papers" were statistically based, only three were solely machine learning (ML) based (11%) with a further two papers including both statistical methods and ML (7%).

4.2. Comparisons of the least cited 2005 papers and most cited papers

The least-cited papers in 2005 are tabulated to investigate:

- The relationship between paper type and citations (see Table 13).
- The relationship between main topic and citations (see Table 14).
- The relationship between main topic and paper type (see Table 15).
- The publication source for conference and journal papers (see Tables 16 and 17).

Compared with frequently cited papers, less cited papers are more likely:

Table 13
Citations and Paper type for least cited 2005 papers.

Number of citations	Number of Journal papers	Number of Conference papers	Total
5	1	0	1
4	1	0	1
3	8	1	9
2	7	3	10
1	7	9	16
0	15	23	38
Total	39	36	75

Table 14
Citations against main topic for least cited papers.

Topic	Number of citations						Total
	5	4	3	2	1	0	
Analysis	0	0	3	1	3	4	11
Development	0	0	2	3	6	7	18
Evaluation	1	0	1	1	2	7	12
Framework	0	1	1	1	1	9	13
Programs	0	0	0	1	0	1	2
Literature survey	0	0	1	0	0	0	1
Use	0	0	0	2	2	6	10
Tool	0	0	1	1	2	4	8
Total	1	1	9	10	16	38	75

Table 15
Main topics against paper type for least cited papers.

Topic	Paper type			Total
	Empirical	Theoretical	Both	
Analysis	6	3	2	11
Development	0	9	9	18
Evaluation	9	3	0	12
Framework	2	9	2	13
Programs	1	1	0	2
Literature survey	1	0	0	1
Use	5	4	1	10
Tool	0	7	1	8
Total	24	36	15	75

Table 16
Source of conference papers for least cited papers.

Conference (or source)	Number of papers
Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR	5
Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS	3
Proceedings – International Computer Software and Applications Conference	3
Proceedings of the ACM Symposium on Applied Computing	2
Lecture notes in AI (vol. 3397)	2
Lecture Notes in Computer Science (vol. 3547)	2
Others (1 paper only)	19
Number of different sources	25

Table 17
Sources Journal papers for least cited papers.

Journal	Number of papers
IEEE TSE	3
Journal of Systems and Software	3
Information and Software Technology	3
Journal of Object Technology	3
Software Quality Journal	3
IEICE Trans on Software Engineering	2
Informatik – Forschung und Entwicklung	2
Jisuanji Gongcheng/Computer Engineering	2
Journal Software Maintenance and Evolution	2
Journal of the Institute of Engineers	2
Ruan Jian Xue Bao/Journal of Software	2
Others (1 paper only)	12
Number of different sources	23

- To be conference papers. i.e. 48% were conference papers compared with 19% for the most cited papers.
- To be framework or tool papers, i.e. 28% compared with 12%.
- To be theoretical papers i.e. 48% compared with 22%.

Compared with frequently cited papers, the results for the less cited papers showed:

- A far wider range of sources for conference papers (including many different volumes in the lecture notes in computer science series) i.e. 25 sources compared with eight for the most cited papers.
- No major source for journal papers i.e. the five most frequently used journals each published three papers.
- The inclusion of foreign language journals (i.e. German and Chinese journals).

Table 18 compares the least cited papers found in 2005 with the most cited papers in terms of various context factors. It must be

Table 18
Context factors in papers.

Context factors	2005 papers cited less than 6 times	Most cited papers
Number of papers	75	100
Object-oriented development	26 (35%)	33 (33%)
Open source	2 (3%)	5 (5%)
Maintenance	7 (9.3%)	27 (27%)

noted that these counts refer only to papers that specified a context. For example, papers that did not state they were maintenance were not necessarily development papers; they were papers for which a maintenance context was not explicitly specified.

It appears that a large proportion of metrics papers whether frequently cited or not relate to object-orientation. The proportion of OO metrics papers found in the study is very similar to the 33% of OO metrics found by Gómez et al. (2008). However, frequently cited papers are more likely to specify a maintenance context (27%) than less frequently cited papers (9.3%). Overall 19.4% of papers were maintenance papers compared with 14% of maintenance metrics found by Gómez et al.

Table 19 identifies the goals of the most cited papers and the least cited 2005 papers. Goals are difficult to categorise and both sets of papers included many papers that were very specific in their goals. However, it seems that most cited papers are more concerned with fault prediction, effort prediction, component fault-proneness classification and re-engineering than less cited papers. Less cited papers were more concerned with metrics programs. The metrics programs category grouped together papers related specifying metrics, collecting, extracting, storing metrics data and evaluating metrics programs.

4.3. Evaluation papers

Although mapping studies tend to stop at categorizing identified papers, the goal of systematic literature review is to attempt to aggregate results into a coherent body of knowledge that answers a specific question. In this section we consider the "Evaluation" papers and consider whether they have potential for aggregation.

Table 19
Goals of metrics papers.

Goal	Most cited papers	Least cited 2005 papers	Total
Fault prediction	12	3	15
Re-engineering	10	2	12
Effort prediction	9	1	10
Component fault classification	8	5	13
Measurement program establishment and evaluation	6	12	18
Sizing	5	1	6
Prediction	4	3	7
Component evaluation	2	0	2
Maintainability	2	2	4
Metrics definition	2	1	3
OS Evaluation	2	0	2
Project management	2	1	3
Quality	2	1	3
Quality prediction	2	0	2
Statistical process control	2	1	3
Test case selection	2	0	2
Design flaw detection	0	3	3
OO metrics development	0	2	2
Project control	0	2	2
Risk management	1	2	3
Testability assessment	0	2	2
Others	27	31	58
Total	100	75	175

Table 20
Summary of metrics evaluation papers.

Metrics being evaluated	Evaluation property		
	Faults	Effort	Size
OO metrics	E3, E4, E6, E12, E16, E25	E1, E2	E1, E6, E13
Web-metrics		E18, E19	
Other code metrics	E14, E15, E20		

Initially, 25 papers were identified as evaluation/validation studies. These papers were read in detail and a common set of information extracted from each, see Table 21. This more detailed assessment suggests that five papers that should be reclassified:

1. Darcy et al. paper (E7) is about the concepts of coupling and cohesion, not about specific metrics.
2. Denaro and Pezzè paper (E8) is about the evaluation of fault-proneness models rather than the metrics on which they rely. It would be better classified as a "Use" study.
3. Briand et al. (E5) is also more about the consistency of fault-proneness models that the specific OO metrics, so could be regarded as a "Use" study.
4. Elbaum et al. (E10) discuss a new metric for test case prioritization, so the paper should be better classified as a "Development" study.
5. Lindvall et al. (E17) used coupling metrics to assess a changed architecture, so their paper should be categorized as a "Use" study.

The remaining papers include six individualistic papers that offer no opportunity for aggregation:

1. Dolado (E9) evaluates whether a novel sizing metric predicts lines of code as well as Mark II function points.
2. El Emam and Birk (E11) validate the ISO/IEC measure of software requirements analysis process capability.
3. Poels and Dedane (E21) present a theoretical paper that considers the application of distance metrics to software measures.
4. Purao and Vaishnavi (E22) present a literature survey aimed at understanding classifying and analyzing ongoing research on object-oriented metrics.
5. Rainer and Hall (E23) discuss factors impacting software process improvement.
6. Schach et al. (E24) present an empirical study of the distribution of maintenance effort across previously suggested categories.

The only thing that can be judged from this set of papers is the wide-range of measurement related issues that require evaluation.

Of the remaining 15 papers, all aim to evaluate a set of metrics against one of three external properties:

- Quality in terms of fault counts, fault proneness (the probability of exhibiting a fault) or fault density (i.e. faults/size).
- Effort either development or maintenance effort.
- Size e.g. lines of code or function points.

The distribution of these papers is show in Table 20. It is clear that a majority of the papers evaluate OO metrics (9 of 14) and a majority of the papers base their evaluation on faults (9 of 15). A total of 6 papers feature OO metrics and a fault-based evaluation. Of the 9 papers using OO metrics, 7 papers used (at least some) of the Chidamber and Kemerer (CK) metrics (Chidamber and Kemerer, 1994).

Table 21
Evaluation papers.

ID	Authors	Year	Evaluation goal	Metrics	Evaluation approach	Evaluation method	OO study	Fault related	Context	Results
E1	Alshayeb, M., Li, W.	2003	Assessing whether OO metrics predict size or effort	CK and related OO metrics	Industrial observational study	Multiple linear regression	Y	N	3 industry systems; 2 agile, 1 long-term evolution	Strong relationships for agile not for long-term products
E2	Bandi, R.K., Vaishnavi, V.K., Turk, D.E.	2003	To evaluate predictive accuracy of three new OO metrics	Interface size; Interaction level; Operation Augment Complexity	Academic experiment	Regression analysis	Y	N	Student subjects performing two maintenance tasks	Metrics predicted effort
E3	Briand, L.C., Wüst, J., Daly, J.W., Victor Porter, D.	2000	To assess whether OO metrics predict faulty classes	67 different OO design metrics including CK metrics, separating import and export coupling	Academic Replicated project study	Logistic regression	Y	Y	8 Student programs (using same functional specification) A total of 113 classes	Best prediction model included 4 coupling measure and 3 inheritance measures. It was better than a size based model
E4	Briand, L.C., Wüst, J., Lounis, H.	2001	To better understand the relationship between OO design metrics and software quality	54 metrics including some CK metrics separating import and export coupling	Academic observational study	Multiple logistic regression	Y	Y	LALO version 1.3 Academic software application. 83 C++ Classes, 40 K LoC. Used by universities and research centers. 6 developers, max 3 in parallel. Fault data for one year, from user-reported failures	The two main factors are size and import coupling
						Principal Component Analysis			Other projects for comparison (UMD)	Cohesion not a good predictor
E5	Briand, L.C., Melo, W.L., Wüst, J.	2002	Can models built on one system predict on another	22 metrics related to coupling, polymorphism, 2 CK metrics; and size	Industrial observational study	Logistic regression; Multivariate Adaptive Regression Spines	Y	Y	2 midsize commercial Java programs; Development; using same technology; same staff (except manager); different design strategies and coding standards;	Cross-project prediction not as good as within project predictions; but better than chance or a simple linear model
E6	Cartwright, M., Shepperd, M.	2000	To assess whether simple OO class metrics predict defect rates or size	9 OO metrics and 3 size metrics	Industrial observational study	Correlation and multiple regression	Y	Y	Subsystem of telecomms product developed with Shlaer-Mellor method; System in use for 12 months; Defects counted from integration	Participation in inheritance hierarchies is associated with higher defect rates Strong relationships between size, states, events and defects with size relationship best
E7	Darcy, D.P., Kemerer, C.F., Slaughter, S.A., Tomayko, J.E.	2005	To assess the relationship between developer understanding and coupling and cohesion and	not applicable	Controlled laboratory experiment	Analysis of variance based on factorial design (Cohesion high and low, Coupling High and low)	Y	N	MSc subjects with programming experience; 2 maintenance tasks per cell; between subject design; academic examples	Significant interaction effect between coupling and cohesions but no main effects
E8	Denaro, G., Pezzè, M.	2002	Define range validity of fault prediction models	38 Module code metrics – size, structure, Halstead metrics	Open Source observational Study	Multivariate Logistic regression (more than 4 faults v 3 or less)	N	Y	versions 1.3 and 2.0 of Apache	Models built on version 1.3 capable of predicting resutls for version 2.0; no real difference between stepwise logistic regression models and PC models. Models better than than size
E9	Dolado, J.J.	2000	To assess whether a component-based sizing method predict conventional size metrics	counts of number of menus; inputs; reports/inquiries; components by type	Academic observational study	Multiple linear regression; neural networks; genetic programming	N	N	46 Academic projects produced using Informax-4GL	Number of components related to LoC; Mark II FP also related to LoC

E10	Emam, K.E., Birk, A.	2000	To assess whether software reliability process assessment predict software and project performance	Capability assessments	Industrial cross-company observational study	Correlation	N	N	SPICE trial data from 29 companies that assessed the reliability process and 56 projects	Requirements process capability impacts productivity for large companies not small, but no relationship with schedule, Budget, customer satisfaction
E11	Elbaum, S., Malishevsky, A., Rothermel, G.	2001	To explore whether an adjusted fault detection rate metric caters for variable test costs and fault severity	Generalized version of APFD metric	Industrial observational study	Simulation studies	N	Y	Small industrial application (SPACE, 6218 lines C)	Metric illustrates issues related to test case prioritization with unequal test costs and fault severity
E12	El Emam, K., Benlarbi, S., Goel, N., Rai, S.N.	2001	To demonstrate the need to evaluate OO metrics after controlling for class size	CK metrics and 2 others	Industrial observational study	Multiple logistic regression	Y	Y	174 classes being reused to develop a telecoms switching system. Post release faults only	Controlling for size removed all relationships between metrics and faults
E13	El Emam, K., Benlarbi, S., Goel, N., Melo, W., Lounis, H., Rai, S.N.	2002	To test whether there is an optimal size for classes	OO size metrics	Multi-system industrial observational study	Logistic regression with threshold	Y	Y	2 systems, two C++ and one Java. C++ Telecomms Evolving system, one release, field faults; C++ telecomms framework being reused for new system field faults; Java New word processor post release faults	No threshold effect in any system
E14	Fenton, N.E.	2000	To evaluate 4 types of hypotheses: Pareto principle; pre v. post release fault data; metrics for fault prediction; benchmarking defect rates	LoC; cyclomatic complexity	Industrial observational study	Pareto-plots; scatter plots	N	Y	2 consecutive releases of a large telecomm system; random selection of new and modified programs	Pre-release fault rates not good predictor of post-release fault rates; LoC poor predictor of pre-release faults and does not predict post-release faults
E15	Graves, T.L., Karr, A.F., Marron, U.S., Siy, H.	2000	To understand the impact of evolution on number of faults	LoC, cyclomatic complexity, Halstead metrics etc. Past faults, changes, age of code, weighted changes	Industrial observational study	General Linear Models; simulation to assess significance	N	Y	Evolution of large legacy system written in C	The best model predicts defect using a weighted sum of changes to model – old changes down graded
E16	Gyimóthy, T., Ferenc, R., Siket, I.	2005	To validate OO metrics for fault prediction	CK metrics including LOCM (both versions) and Loc	Open Source observational study	Linear regression, Logistic regression, machine learning (decision trees and neural networks)	Y	Y	Mozilla (1.6 excluding classes bug-free in all versions) and Bugzilla (excluding bugs before Mozilla 1.0 and after 1.7)	All variables except NOC were significant. CBO performed best
E17	Lindvall, M., Tvedt, R.T., Costa, P.	2003	Evaluation of architecture based on metrics	Coupling between and within modules (not standard OO metrics)	Comparison of metrics values for two versions of an architecture	Simple counts and histograms for each architecture	Y	N	Two versions of a research prototype V1 built by two people is 10 K Java. V2 built by distributed team of 7 people is 15 K Java with 22 new requirements	The new version appears better structured and should be more maintainable.
E18	Mendes, E., Mosley, N., Counsell, S.	2002	To investigate which types of web metrics best predict development effort	Web metrics of three types, size, complexity, functionality	Academic observational study	Multiple linear regression	N	N	Web application student projects aimed at teaching .43 CS students stated the work, 37 results were included in final analysis.	No statistical difference was found between models based on each of the three metric types
E19	Mendes, E., Mosley, N., Counsell, S.	2005	Evaluation of Web sizing metrics	Industry-based surveys and multi-project multi-company – observational studies	Multi-company industrial observational study	Multivariate step-wise regression analysis	N	N	Many companies in many countries working on web application development	Number of web pages and functionality measures were best

(*continued on next page*)

Table 21 (*continued*)

ID	Authors	Year	Evaluation goal	Metrics	Evaluation approach	Evaluation method	OO study	Fault related	Context	Results
E20	Nagappan, N., Ball, T.	2005	Evaluation of code change measures for defect prediction	Changed, deleted code size, files changed, weeks being changed, number of changes	Industrial observational study	Stepwise regression. Split data set to validate predictions (using actual v. estimate correlation)	N	Y	Windows Server 2003	Relative code churn related to defect rate; Relative values better than absolute values;
E21	Poels, G., Dedene, G.	2000	To identify necessary and sufficient axioms for distance-based software measurement	Distance-based software metrics	Axiomatic	Theoretical	N	N	N/A	When applicable, the distance approach is generic, flexible and formal. It permits both metric validation and construction
E22	Purao, S., Vaishnavi, V.	2003	To classify all OO product metrics	OO product metrics	Theoretical framework and Literature review	Metrics mapped to framework	Y	N	N/A	See Section 2
E23	Rainer, A., Hall, T.	2003	To assess which factors affect the success of software process improvement	26 possible success factors	Multi-company survey and interviews 4-comopany case study	Textual analysis	N	N	Survey of 84 companies from 13 countries. Interviews from 13 of the companies. Group interviews in 4 companies.	Case study and survey found 7 SPI related factors: Executive support, experienced staff, internal process ownership, metrics, procedures, reviews and training. Two factors not related: reward schemes and estimating tools)
E24	Schach, S.R., Jin, B., Yu, L., Heller, G.Z., Offutt, J.	2003	To assess the distribution of maintenance across standards categories	Maintenance activities at different levels of abstraction (line, change-log, module, program) classified using the Lientz, Swanson and Tompkins categories	Industrial observational case study	Tabulation of maintenance against maintenance categories compared with LST distribution	N	N	10 original files plus 138 modified file of Commercial real-time product, 39 versions of Linux kernel, versions 2.4.0 to 2.7.2.3 of GNU Complier Collection	Found far more corrective and far less adaptive and perfective maintenance than LST survey results
E25	Subramanyam, R., Krishnan, M.S.	2003	Can specific metrics predict number of defects	CK metrics: CBO, DIT, WMC, LoC	Industrial observational study	Multiple Regression using 1/(1+defects) as outcome variable	Y	Y	Software development laboratory producing commercial apps; One large B2C e-commerce application using both C++ and Java; defects counted from field and customer acceptance testing	Size weighted model suggests size, WMC, DIT (inverse) and interaction between DIT and CBO related to defects for C++ and size and DIT and CBO (inverse)

Although nine papers evaluated metrics by looking at relationships with faults, only one paper (E16) was selected by Catal and Diri in their mapping study of fault prediction studies (Catal and Diri, 2009). This reflects the different scope of the papers: Catal and Diri are focused on the analysis methods used to construct fault models, whereas this set of papers focuses on using fault models to validate metrics. However, Catal and Diri did not include Briand et al. paper (E5) nor Denaro and Pezzé's paper (E8), both of which were reclassified as a fault modeling papers. It is also notable that only one paper (E6) published its dataset. This is a worse record for data publication than reported by Catal and Diri (2009) who found that 31% of 78 papers used public data sets. A more detailed discussion of evaluation issues arising from the papers is presented in Section 5.2.

5. Discussion

5.1. What influences the metrics community?

The study suggests that the metrics community is influenced primarily by journal papers with an empirical content, particularly papers published in TSE. Metrics research is not dominated by object-orientation but a reasonably large proportion of papers (approximately one third) are OO related.

The least cited 2005 papers indicate that theoretical papers published in conferences have little impact on other researchers. Indeed many were not even cited by their own authors. These results are consistent with the results reported by Jørgensen and Shepperd (2007) who found that cost estimation studies usually cited only papers from a restricted number of sources. In the case of cost estimation, Jørgensen and Shepperd regard this as a problem because relevant research might be being overlooked. In the case of software metrics, it is theoretical studies of new metrics, frameworks and tools that are being overlooked, which may be less of a problem.

5.2. Empirical validation studies

Empirical validation studies have a strong influence on the metrics community, but a detailed review of the most influential evaluation papers shown in Tables 21 and 20 raises a number of issues that are discussed below. In this section we attempt to integrate the results for the papers included in Table 20 excluding papers E18 and E19 which evaluate Web-based size measures for effort estimation and, as such, are rather different to the other papers.

5.2.1. Invalid empirical validations

Hitz and Montazeri (1996) demonstrated clearly that both versions of the Lack of Cohesion Metric (LOCM) proposed by Chidamber and Kemerer are theoretically invalid. In my opinion the Coupling Between Objects (CBO) metric is also theoretically flawed because it treats forward and backward links as equivalent. My own research confirmed that fan-in and fan-out metrics have very different relationships with faults in procedural systems (Kitchenham et al., 1990) and I found similar results when investigating forward and backward coupling in an OO system (Wilkie and Kitchenham, 2001). Briand and his colleagues seem to have a similar view because they always separate input and export coupling metrics (E3, E4, E5) and do not use the CBO metric.

Proving that a metric is theoretically invalid means that empirical validation is unnecessary, in fact attempting to empirically validate an invalid metric is itself invalid. However, as shown in Table 22 only one of the studies validating CK metrics used neither the CBO nor LOCM (based on the CK counting procedure).

Table 22
Use of LOCM and CBO metrics.

Paper Id	LOCM	CBO
E1	Yes	No
E3	Yes	No
E4	Yes	No
E5	No	No
E12	Yes	Yes
E16	Yes	Yes
E25	No	Yes

Any empirical results obtained from invalid metrics will be meaningless (i.e. totally arbitrary), so invalid metrics should be removed from metrics suites and no further "empirical validation" studies including such metrics should be published. Thus, these results raise the questions of why researchers are wasting time "validating" invalid metrics. Some potential reasons can be derived from the papers:

- **Because CK metrics are widely used**. Alshayeb et al. (E1) note some of the problems with CK metrics. They do not use the CBO metric using instead two alternative coupling metrics. Although they reference Hitz and Montazeri (1996), they still use the standard LOCM metric. They state they have used CK metrics because they are widely accepted by the software engineering community.
- **For completeness.** Briand and his colleagues (E3, E4) do not use the CBO metric. They use the CK LOCM metric as one of many different "lack of coherence" metrics.

Other papers that use LOCM make no reference to problems with the LOCM metrics. Thus, overall it appears that researchers either ignore theoretical validation results or do not believe them.

5.2.2. Relationships with size

In 1997, Rosenberg made two interesting points about models including lines of code (LoC) Rosenberg (1997):

1. The negative relationship between defect density and LoC observed in several metrics papers is an artifact. There will be a negative correlation between size and fault density (i.e. fault count/lines of code) for components even if fault counts and lines of code are independent because of the functional relationship between the variables. This is true for any metric not just LoC and suggests that models of fault density may be misleading if size is included.
2. LoC is most useful as a covariate in fault models not as a predictor in its own right. This is true for any size metric not just LoC.

These points relate to three of the frequently cited evaluation papers:

- Point 1 casts some doubts on the results presented in E20 which uses both defect rates and LoC change rates i.e. correlated x/z and y/z.
- In E13, El Emam et al. demonstrates empirically that point 1 is also true for OO systems, by showing that in three separate OO systems, there was no class size for which smaller classes were more fault prone than larger classes.
- In E12, El Emam et al. criticise previous empirical validation studies of OO metrics because they have not considered whether results are confounded with size. Using logistic regression they demonstrate that relationships between OO metrics and faults disappear in a large OO system after allowing for the relationship with size.

Evanco argues that LoC cannot be a confounding factor because unlike class design metrics it can only be measured during or after implementation (Evanco, 2003). This argument appears semantically valid. However, with respect to evaluation of class metrics, Evanco's argument would imply that design metrics must be validated only with respect to design-related faults not code-related faults. Furthermore although design metrics may not be confounded with code metrics, they can still be *surrogate* measures of code size. Thus design metrics may directly predict design faults and indirectly predict code faults. In addition, if we expect design metrics to predict design faults, we must consider whether or not such predictions are affected by a development process that incorporates design inspections in which many design faults will be detected before implementation and so will not be observed during testing.

Of papers evaluating CK metrics against faults published after 2002 (when E12 was published), E16 did not make any adjustment for size, E25 included size in its weighted least squares analysis. Among papers published between 2000 and 2002, E3 and E6 both found strong correlations between OO metrics and size but do not include LoC in their multivariate models. Paper E4 explicitly states that in their data set class metrics "capture an additional effect beyond size" based on design size metrics such as "number of parameters". However, they do not report whether or not design inspections were used. In addition, *none* of these papers consider whether the faults were code or design faults.

The importance of Rosenberg's paper should not be underestimated. It is sometimes not clear that researchers understand the implication of demonstrating a linear relationship between a size metric and number of faults (or fault-proneness). Finding a linear relationship between size (e.g. line of code) in a component and number of faults in a component **does not** validate component size as a predictor of faults because it is consistent with the null hypothesis that each line in the system as a whole has the same probability of exhibiting a fault (or equivalently that the distribution of faults across lines of code is completely random). This explains why some researchers consider size an "exposure" measure and propose using Poisson or negative binomial regression to model the relationship between faults and size (Mayer and Sykes, 1989).

5.2.3. Contradictory results

In E14, Fenton and Ohlsson discuss relationships between faults and size in a large evolving telecommunications system. They found

- Pre-release fault rates were not good predictor of post-release fault rates.
- LoC was a poor predictor of pre-release fault counts.
- LoC did not predict post-release fault counts.

These results seem to seriously contradict Rosenberg's results (Rosenberg, 1997) and those of many other researchers. However, Fenton and Ohlssons' results relate to specific release of a very large evolving system. They selected a mixture of new and amended modules and did not appear to make any specific adjustment to their results to allow for reused code in amended modules. My own experience is that relationships between new and amended programs are very different. In a large evolving product, I found quite linear relationships between size and defect counts for programs in a new subsystem (albeit with a large outlier), but a very much more complex pattern in a subsystem including both some amended and some new programs (Kitchenham, 1984).

In E15, Graves et al. looked at predicting faults based on the amount of code that was changed in an evolving system. They found a model that weighted recent code changes more than old changes gave the best predictions.

Alshayeb and Li reported related results in a study of OO metrics and size or maintenance effort (E1). They found OO metrics could predict lines added, deleted and changed in short-cycles agile process but could not predict the same aspects in a long-cycled framework process.

A linear model between code size and code faults is likely to hold for modules that were developed in the same way and have had the same opportunity to manifest any faults. This is possible for the first release of systems elements (classes, or modules) assuming that they have been inspected according to their size and tested according to coverage criteria (such as lines of code or control flow which is also related to size). A lack of correlations between pre-release and post-release faults would also be expected under these conditions (not only because of the issue of truncating the fault distribution) but also because coverage-based testing would be replaced by usage implying different exposure patterns for faults that might be independent of module size.

The underlying statistical reason why we can expect contradictory results analyzing component metrics from evolving systems is that as components age and change at different rates they become less comparable with one another. Most statistical tests assume that measures are independently and identically distributed (i.i.d.). However, defect counts from components that have been introduced to a system at different times and changed at different times will almost certainly violate the i.i.d. assumption, rendering most statistical analysis methods problematic. Basically, when dealing with evolving systems metric "snap-shots" are likely to be misleading. In my experience, design-based metrics are more stable than code in evolving systems, so object-oriented design metrics will become less and less useful for fault predictions as a system evolves and code change metrics are likely to be the most valuable predictors. Nor would I expect machine learning techniques to work any better. They have their own problems such as finding spurious patterns in randomly generated data (Shepperd and Kadoda, 2001).

Overall papers E1, E14, E26 are consistent with the following conclusions:

- Code metrics extracted at a specific point in time are unlikely to predict fault rates well in evolving system.
- Code *change* metrics (and other component history related factors) are likely to predict fault rates in an evolving system better than simple snap-shot based metrics.
- Relationships between OO design metrics and external factors (code changes and maintenance effort) will decay over time. This is also supported by a more recent study (Olague et al., 2007) which concludes "for highly interactive or agile systems the metrics will not remain effective for ever".

5.2.4. Too many comparisons

One major problem with aggregating results is that there are many possible OO metrics and it is difficult to assess whether there are reliable trends if researchers analyze many different metrics in the same study. For example E3 uses 66 different OO metrics (plus a count of number of statements) and E4 uses 54. Both studies apply principal component analysis to organise the large number of metrics but using complex multivariate methods makes assessing the results even more difficult. Furthermore using a large number of variables raises an additional analysis problem since the more variables you test for correlations, the more likely you are to find some significant correlations by chance (Courtney and Gustafson, 1993).

5.2.5. Conclusions for empirical validation

Given that OO metrics, in particular the CK metrics have been subjected to empirical validation, since they were initially proposed (see Li and Henry, 1993; Basili et al., 1996), it is a little surprising that there have been so many empirical validations and that they are still continuing (e.g. Olague et al., 2007). How many empirical evaluations are needed to decide whether a set of metrics are valid or not?

"Empirical validation" by looking for relationships between metrics and faults in specific data sets does not seem effective. In my opinion, one underlying problem is that empirical validations are treated as if they are experiments whereas they are closer to embedded case studies (Yin, 2003). In case studies, the nature of the case, i.e. the situation in which the data set is collected, plays a critical part in the explanation of outcomes. In the case of the fault-related empirical validation studies discussed above, context issues include:

- The development process, in particular the inspection and testing processes.
- The state of the application being measured, in particular whether it is a new development or a system undergoing continuous enhancements.
- In evolving systems whether metrics are snap-shots or measures of the changes made to size and structure over a particular time period.
- The type of faults detected i.e. whether design or code.

All these issues need to be considered if appropriate hypotheses (or propositions if we use Yin's case study terminology (Yin, 2003) can be specified. Furthermore, in a good quality case study:

- Detailed propositions are specified before data analysis takes place.
- Multiple-case case studies specify in advance whether cases are literal replications (i.e. ones where the same outcome expected) or theoretical replications (i.e. ones where a different outcome is expected).

Currently many studies use multiple data sets. However, most researchers still construct post-hoc explanations for different outcomes rather than refining their hypotheses to predict differences. Overall, we need to reflect more on what we mean by "empirical validation" and identify appropriate methods for performing such studies. The complexity of the interaction between context and outcomes suggest support for Fenton and Neil's proposals to model our understanding of relationships among software factors and outcomes such as fault proneness in much more detail (Fenton and Neil, 1999).

Post-hoc data correlation and regression studies say nothing about causality (and neither do post-hoc machine intelligence studies). If we want to evaluate causality between metrics and software development properties, we need to do more formal experiments, such as Bandi et al. study of three OO complexity metrics (E2) and Darcy et al. study of the impact of cohesions and coupling on developer understanding (E7). The actual results of the E7 study were puzzling since they suggested that high coupling and low cohesion artifacts (i.e. artifacts bad in two dimensions) were easier to understand than low coupling and low cohesion artifacts (i.e. artifacts bad in only one dimension), nonetheless such research is important and needs to be replicated with different materials.

It is important that we understand development concepts such as cohesion and coupling and how to measure them, because many of the "Use" papers are re-engineering papers, which advocate changing the design of applications to optimize various "metrics". We need evaluation studies that investigate whether our metrics are a reliable basis for re-engineering.

5.3. Keywords and abstracts

In previous papers (Budgen et al., 2008; Kitchenham et al., 2008), my colleagues and I have suggested that to support secondary studies like this one, abstracts need to be improved and that empirical studies at least should adopt structured abstracts. Overall I found the abstracts reasonably well-written but it was sometimes difficult to determine what the real purpose of the paper was. I would recommend researchers to remember to have a sentence on the purpose of the paper in the abstract and include terms related to their goals in the keywords.

A major problem I found was the number of false positives in the search results. Precision would have been better if I had been able to rely on "software metrics" or software measurements" rather than treating "software" and "metrics or measurements" as separate terms. However, 61% of the most cited papers did not use the term "software metrics" or "software measurement" in their keywords. Many papers instead used "OO metrics", "metrics", "product metrics", "process metrics", or some specific metric name. Curiously, 10 papers included the keyword "computational complexity" which is clearly inappropriate (computational complexity being about whether or not an algorithm will complete processing in a finite period of time).

It may be too late for the community to agree on appropriate keywords for metrics papers but in future I will attempt to use keywords related to:

- The goal of the study.
- The main and secondary topics of the study including the term "software metrics" explicitly.
- The name of any specific metrics used in the study.
- The general methodology such as empirical study, literature review, or theory used by the paper and the specific methods used.

I recognise, however, that this approach to keywords is inconsistent with the ACM categories that are being adopted by many computer science journals.

5.4. Limitations

There are many limitations to this study. From the viewpoint of the search process:

- It is based on a single indexing system (SCOPUS). However, SCOPUS is particularly useful because it indexes publications from a large number of publishers including ACM, IEEE, Elsevier, and Springer.
- The search was based on abstract, title and keywords only and clearly missed topics that might normally be considered to be software metrics. The obvious omission was function point analysis which requires a search string including the terms "function points" or "fpa".
- The main searches were based on a restricted number of years. The selection of years was based on the assumption that papers published between 2000 and 2005 would be the ones most likely to be currently impacting the research community.
- The search process was performed by a single person implying that some papers might have been included or excluded incorrectly. The test–retest figures suggested that papers were likely to have been omitted, and the search for most cited papers was repeated.

From the viewpoint of data extraction, a single researcher represents a threat to the validity of the data. Test–retest suggested that the simple classifications were relatively reliable. However, it is clear from the results of the detailed examination of papers classified as "evaluation studies", that there were classification errors that could only be detected by detailed reading of the individual papers. This implies there are many classification errors remaining in the other categories. In addition, the classification of goals was particularly difficult and would have benefited from input from other researchers.

6. Conclusions

In spite of the limitations of this study, I believe it is still useful, because the most important function of a mapping study is to provide classified references that can act as the *starting point* for more detailed studies (primary studies as well as secondary studies) and this paper:

- Identifies the most influential papers (in terms of citations) from 2000 to 2005 (according to SCOPUS).
- Compares papers that are frequently cited with those that are not.
- Provides a critical assessment of papers classified as "Evaluation" studies.

It seems that empirical validation papers are of major importance to the software metrics research community. However, this study suggests that the limitations of empirical studies are not always understood, in particular, the impact of the context in which data arises, the impact of constructed metrics (such as defect density, change density), the implication of linear relationships, and the notion of measures needing to be identically, independently, distributed. Furthermore, we appear to have no idea when to stop empirical validation, even in the case of well-investigated metrics such as the Chidamber–Kemerer metrics.

I believe the software engineering research community must take up the challenge to aggregate its results into an empirically-based body of knowledge. The evidence from this mapping study and the others described in Section 2 is that there is a wealth of software metrics studies available. This study indicates that the results from some primary studies seem to be capable of being integrated in order to explain contradictory results, however, there appear to be several problems in moving from mapping studies to more detailed systematic literature reviews:

- Evidence-based practice requires that all evidence be critically appraised in order that practice is based on good quality evidence. An important part of any systematic literature review is therefore to evaluate the quality of primary studies included in the review. The results in this study suggest that the metrics community have a tendency to develop some methodological approach to an issue (in this case empirical evaluation using regression models based on industry data sets) and then continue to perform many such studies without reflecting on whether the methodology is appropriate. It is important that we understand our empirical methods well-enough to be able to assess correctly whether the results of empirical metrics papers can be trusted.
- Failure to reflect on the impact of using industry data sets for metrics evaluation suggests that the critical impact of context on study outcomes has not been fully appreciated. However, without context information, the results of empirical studies are unlikely to address industry-related questions.
- The results from this study suggest that metrics studies currently overlook previous relevant research such as Rosenberg's study (Rosenberg, 1997), perhaps because researchers assume that behaviour of OO metrics is fundamentally different from behaviour of procedural metrics. Metrics researchers need to understand the difference between statistical properties (i.e. the problem of density metrics), and conceptual properties (i.e. those related to measurable attributes) which are to a large degree independent of the specific coding paradigm, and paradigm specific properties such as the specific way in which attributes are measured.

Finally, we must expect increasingly to use tools such as SCOPUS to support our literature searches. However, it would assist us all, if we could improve our abstracts and key-wording to improve the accuracy and precision of automated searches.

Acknowledgement

This study was funded by the UK Engineering and Physical Sciences Research Council Project EP/E046983/1.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jss.2009.06.041.

References

Abran, A., Moore, J.W., Bougque, P., Dupuis, Trip, L.L. (Eds.), 2004. Guide to the software engineering body of knowledge. IEEE Computer Society Press.
Basili, V., Briand, L., Melo, W., 1996. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering 22 (10), 751–761.
Bellini, C.G., Pereira, R.D.C.D.F., Becker, J.L., 2008. Measurement in software engineering from the roadmap to the crossroads. International Journal of Software Engineering and Knowledge 18 (1), 37–64.
Briand, L.C., Morasca, S., Basili, V.R., 2002. An operational process for goal-driven definition of measures. IEEE Transactions on Software Engineering 28 (12), 1106–1125.
Budgen, D., Kitchenham, B.A., Charters, S.M., Turner, M., Brereton, P., Linkman, S.G., 2008. Presenting software engineering results using structured abstracts: a randomised experiment. Empirical Software Engineering 13 (4), 435–468.
Catal, C., Diri, B., 2009. A systematic review of software fault prediction studies. Expert Systems with Applications 36, 7346–7354.
Chidamber, S.R., Kemerer, C.F., 1994. A metrics suite for object-oriented design. IEEE TSE 20 (6), 476–493.
Courtney, R.E., Gustafson, D.A., 1993. Shotgun correlations in software measures. Software Engineering Journal 8 (1), 5013.
Dybå, T., Kitchenham, B.A., Jørgensen, M., 2005. Evidence-based software engineering for practitioners. IEEE Software 22 (1), 58–65. January.
Evanco, W.M., 2003. Comments on "The confounding effect of class size on the validity of object-oriented metrics". IEEE Transactions on Software Engineering 29 (7), 670–672.
Fenton, N.E., Neil, M.A., 1999. Critique of software defect prediction models. IEEE TSE 25 (5), 675–689.
Fenton, N.E., Pfleeger, S.L., 1997. Software Metrics: A Rigorous and Practical Approach, second ed. PWC Publishing Company.
Glass, R.L., Ramesh, V., Vessey, I., 2004. An analysis of research in computing disciplines. CACM 47 (6), 89–94.
Gómez, O., Oktaba, H., Piattini, M., García, F., (2008). A systematic review measurement in software engineering: state-of-the-art in measures. ICSOFT 2006, CCIS 10, LNCS 5007, pp. 165–176.
Hitz, M., Montazeri, B., 1996. Chidamber and Kemerer's metrics suite: a measurement theory perspective. IEEE TSE 22 (4), 267–271.
Jørgensen, M., Shepperd, M., 2007. A systematic review of software development cost estimation studies. IEEE TSE 33 (1), 33–53.
Kitchenham, B.A., 1984. Program history records: a system of software data collection and analysis. ICL Technical Journal, 103–114. May.
Kitchenham B.A., Charters S.M., (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3. EBSE Technical Report EBSE-2007-01, Keele University and Durham University.
Kitchenham, B.A., Pickard, L.M., Linkman, S.J., 1990. An evaluation of some design metrics. Software Engineering Journal 5 (1), 50–58.
Kitchenham, B.A., Dybå, T., Jørgensen, M., (2004). Evidence-based software engineering. In: Proceedings of the 26th International Conference on Software

Engineering, (ICSE '04), IEEE Computer Society, Washington DC, USA, pp. 273–281, ISBN: 0-7695-2163-0.
Kitchenham, B.A., Brereton, O.P., Owen, S., Butcher, J., Jefferies, C., 2008. Length and reliability of structured software engineering abstracts. IET Software 2 (1), 37–45.
Li, W., Henry, S., 1993. Object-oriented metrics that predict maintainability. Journal of Systems and Software 23, 111–122.
Mayer, A., Sykes, A., 1989. A probability model for analysing complexity metric data. Software Engineering Journal 4 (5), 254–258.
Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S., 2007. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Transactions on Software Engineering 33 (6), 402–419.
Purao, S., Vaishnavi, V., 2003. Product metrics for object-oriented systems. ACM Computing Surveys 35 (2), 191–221.
Rosenberg, J., 1997. Some misconceptions about lines of code. In: Proceedings of Fourth International Software Metrics Symposium, Metrics '97, pp. 137–142.
Shaneck, M., Mahadevan, K., Kher, V., Kim, Y., 2005. Remote Software-based Attestation for Wireless Sensors, Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3813 LNCS, pp. 27–41.
Shepperd, M., Kadoda, G., 2001. Using simulation to evaluate prediction techniques. In: Proceedings International Software Metrics Symposium, pp. 349–359.
Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.K., Rekdal, A.C., 2005. A survey of controlled experiments in software engineering. IEEE TSE 31 (9), 733–753.
Wilkie, F.G., Kitchenham, B.A., 2001. An investigation of coupling, reuse and maintenance in a commercial C++ application. Information and Software Technology 43, 801–812.
Yin, Robert K., 2003. Case Study Research: Design and Methods, third ed. Sage Publications.

Barbara Kitchenham is Professor of Quantitative Software Engineering at Keele University in the UK. She has worked in software engineering for over 30 years both in industry and academia. Her main research interest is software measurement and its application to project management, quality control, risk management and evaluation of software technologies. Her most recent research has focused on the application of evidence-based practice to software engineering. She is a Chartered Mathematician and Fellow of the Institute of Mathematics and Its Applications, a Fellow of the Royal Statistical Society and a member of the IEEE Computer Society.