Metrics and Statistical Techniques Used to Evaluate Internal Quality of Object-Oriented Software: A Systematic Mapping

Mariana Santos
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
mariana@bsi.ufla.br

Paulo Afonso Junior
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
pauloa.junior@dcc.ufla.br

Paulo Henrique Bermejo
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
bermejo@dcc.ufla.br

Heitor Costa
Department of Computer Science
Federal University of Lavras
Lavras, Brazil
heitor@dcc.ufla.br

Abstract — Efficient ways to measure and evaluate internal quality of object-oriented (OO) software are widely discussed in the literature. Studies in software engineering has used statistical techniques/models to evaluate quality in different aspects (quality characteristics and sub characteristics) and using different variables (traditional and object-oriented software metrics) to quantify it. This paper presents results of applying a systematic mapping to identify the most software metrics cited and the most statistical techniques/models used to evaluate quality in OO software. As a result, we highlighted 79 papers related to the research question in a total of 8,231 papers published in scientific journals. The main contributions of this work are (i) a list of the most used metrics and measurable properties, (ii) a list of the most used statistical techniques/models, and (ii) the main references, themes to be explored and recommendations for future research.

Keywords — Global Software Development, Software Project Management;

I. INTRODUCTION

Although the object-oriented (OO) technology is consolidated in the software development, efficient ways to measure and evaluate the internal quality of OO software is still discussed in the literature [3]. Quality is defined as the degree to which characteristics of products or services satisfy explicit and implicit needs of stakeholders, adding value to products or services [9]. Internal quality is the totality of the software characteristics in an internal point of view. It is measured and evaluated in relation to the source code quality through software metrics [9] [15]. In ISO/IEC 25010, the internal quality is part of the model for software product quality [9].

Software metrics is a scale used to categorize qualitative data from software and its internal and external specifications [9]. They can be collected throughout the development process by identifying important aspects of a project, for example, how much effort, cost, complexity, and, when specific, represent inherent characteristics to adopted software engineering technology, such as the inheritance (OO) [18].

Studies of software engineering area have focused on understanding how metrics can characterize the quality of software [2]. Among the sub-topics covered are fault-prone factors [1] [6], productivity or maintenance effort [7] [8], software reuse, and software refactoring [17]. In these assessments, various models for measuring OO software quality have been proposed [14], which results are obtained using techniques for inferential analysis and multivariate data analysis. Although the studies goals are many, we still know very little about what metrics are more relevant and which techniques are most appropriate to get better results to characterize the software quality or a particular aspect, for example, maintainability. A systematic mapping identifies these studies with their goals and results which aims to identify and interpret available studies and relevant to a specific research question. This research methodology has been widely used in recent years in software engineering area.

In this paper, the aim is to identify studies that evaluate the internal OO software quality, using technical/statistical models. We found 79 papers and the results showed 15 main metrics used (e.g., Lack of Cohesion between Objects - LCOM had 40 citations), five ISO/IEC 25010 quality characteristics addressed (Maintainability, Functional Suitability, Security, Usability, and Reliability were cited), and 10 main statistical techniques/models used to evaluate the internal quality of software (e.g., Descriptive Analysis had 42 citations).

The paper is organized as follows. Related work to the research is presented in Section II. The protocol and the methodology are shown in Section III. The result of data extraction is analyzed in Section IV. Conclusions, contributions and suggestions for future work are discussed in Section V.

II. RELATED WORK

In the literature, some studies with related themes used Systematic Literature Review (SLR) technique. In one study [16], SLR was used to identify the use of metrics, methods, and techniques to predict software maintainability. In this study, the results were (i) the size, complexity, and coupling metrics are mostly used as maintainability predictors and (ii) the necessity of building robust, accurate, and reliable models; those models must be non-specific to a software domain.

978-1-5090-3339-3/16/$31.00 ©2016 IEEE

In another study [11], SLR was used to identify whether the CK metrics are good predictors of functional adequacy in OO classes. The WMC, CBO, RFC, and LCOM metrics were considered good predictors, specifically to functional correctness sub characteristics. Finally, 20 studies have been analyzed. The third study [3] carries out a SLR on the effectiveness of OO technology. As a technical result, 138 studies have been analyzed and identified that most empirical studies is focused on identifying metrics. Relatively, few studies considered the efficacy in evaluating the quality of OO software.

In the fourth study [10], SLR was used to identify which metrics are useful for quantitative analysis of OO source code quality and the connection of such metrics with external quality attributes, such as, maintainability, reliability, and functionality. However, this study does not provide the link between metrics and statistical techniques most commonly used for this purpose. Another paper [5] has proposed a critical analysis of the OO metrics. As results, the WMC, NOC, RFC, and CBO metrics are relevant and the Lorenz and Kidd suite is not relevant for evaluating software quality.

This study differs from previous ones by focusing on a mapping study of the most used metrics and their relationships to characterize the internal quality of software and what are the statistical techniques/models used to describe and estimate these relationships. Thus, the study goes further to establish links between the relevant metrics, but seeks to find topics that have/need been discussed in future studies related to internal quality of software. Another important point is not limited to specific characteristics and properties to which the metrics are related. In addition, the study focuses on analyzing only studies in the past decade, although important to understand the current scenario of studies into internal quality of software, previous studies in the decade are already addressed in previous reviews.

III. Methodology

We used the phases of SLR for carrying out our systematic mapping. SLR it is a technique for identifying, evaluating, and interpreting available relevant studies or research for a specific research question, subject area or phenomenon of interest [12]. The phases are: i) **Planning phase**: The reason for performing of SLR is established; ii) **Execution phase**: The research on sources defined in the previous phase is performed. The papers study and classification can be made, guided by the inclusion and exclusion criteria; and iii) **Analysis Phase**: The collection and organization of the data extracted from the papers selected. The result is analyzed globally, in order to get conclusions about the theme chosen or improve the previous definitions on the protocol, if necessary.

A. Research Questions

The aim is to identify studies that evaluate the software internal quality of OO software, using statistical techniques/models. The following research questions were formulated:

```
Q1: What are the main metrics
(traditional and object-oriented)
investigated/used, to evaluate the
software internal quality?

Q2: What are the main statistical
techniques/models used by researchers at
the software engineering area to evaluate
the internal quality software?
```

B. Selection of Studies

We selected as sources, search engines on the web, aimed at extensive search of full scientific texts and metadata and we conducted research as following: i) engines should contain the advanced search option with use of keywords; ii) results should be filtered by publication year, area and/or publication type; and iii) query results should be exported in BibTex format. These engines should present invariance in the search result when using the same set of keywords; thus, we chose IEEE (http://ieeexplore.ieee.org/), Scopus (http://www.scopus.com), ScienceDirect (http://www.sciencedirect.com), Springer (http://link.springer.com/advanced-search), and Ei Compendex (http://www.engineeringvillage.com/).

Among search sources, ACM Digital Library was considered to be used as a repository. However, it presented unsatisfactory navigation and did not provide option to export query results. In a study on search engines [4], it was found that the results provided by the ACM search engine are inconsistent in some situations. The Springer's search engine provides option to export result, but the exportation of some documents have flaws like lack of abstract and empty files. To perform the search, we used the search string built based on the keywords and synonyms defined in the study:

```
(software OR application OR applications
OR system OR systems OR program OR
programs OR "software system" OR
"software systems")

AND

(metric OR metrics OR "software metrics"
OR "code metrics" OR "object-oriented
metrics" OR measure OR measures OR
measurement OR measuring)

AND

(quality OR "internal quality" OR "code
quality" OR "software quality") AND
("object oriented" OR "object-oriented"
OR "oo")

AND
```

```
("statistics" OR "statistical analysis"
OR "statistical analyses" OR "statistical
technique" OR "statistical techniques" OR
"statistical approach" OR "statistical
approaches" OR "statistical tools" OR
"statistical method" OR "statistical
methods" OR "statistical model" OR
"statistical models" OR "quantitative
analysis")
```

As inclusion criteria, we considered that paper must belong to computer science area, be published in Journal or Proceedings (Conference Paper), and be published in English. Besides, paper must present a study on the evaluation of internal quality of software using OO/traditional metrics and statistical techniques/models. As exclusion criteria, we disregarded studies that are restricted or incomplete text, in press papers (unpublished effectively), non-papers (e.g., Table of Contents, Index e Standards), and does not meet inclusion criteria. Four researchers (RA, RB, RC, and RD) have been involved in the selection, using the following procedure:

1. RA executed the search string in the selected sources, documenting the results on JabRef (http://jabref.sourceforge.net/);

2. RA found and deleted non-papers and repeated papers. For repeated papers, RA kept the paper whose keywords best had described the main subject;

3. RA and RB evaluated the papers considering inclusion and exclusion criteria, documenting the results. This evaluation was conducted by reading title, abstract, and keywords. Disagreements were resolved between RA and RB. In case of non-consensus, the paper was included. The excluded papers were documented in a justification list;

4. RC evaluated the papers obtained from the search string considering title, abstract, and keywords. After, RC conducted the intersection between the selected papers by him/her and the selected papers by RA and RB. RA, RB, and RC resolved discrepancies. The final result was a set of papers;

5. RD evaluated the set of papers obtained and has resolved discrepancies by consulting RA, RB, and RC. The final result was a set of papers resulting for full reading;

6. The selected papers were read and data were extracted. Papers that did not met the inclusion criteria were discarded.

We used an inter-range agreement (kappa) to evaluate the reliability of the researchers' evaluation. To verify the agreement among the researches, the papers found were divided in two categories: i) included, papers that fulfill all the inclusion criteria; and ii) excluded, papers that do not meet at least one of the inclusion criteria or papers that fulfill all the exclusion criteria. As we have more than two raters (researchers), we used Fleiss' Kappa and a significance test (p-value) to evaluate if the agreement among the researchers is reasonable. The two hypotheses tested were:

```
H0: There is no agreement among the
    researchers (kappa = 0);

H1: There is agreement among the
    researchers (kappa > 0).
```

In Table I, we presented reference values for Kappa's test [13]. For example, for value between [0.20; 0.39], we have a Fair agreement among the researchers. We used an on-line application called Kappa Agreement's Analysis to calculate the Kappa's values and the statistical significance test.

TABLE I. KAPPA'S REFERENCE VALUES

Kappa's Values	Interpretation
< 0	No agreement
0 - 0.19	Poor agreement
0.20 - 0.39	Fair agreement
0.40 - 0.59	Moderate agreement
0.60 - 0.79	Substantial agreement
0.80 - 1.00	Almost perfect agreement

IV. RESULTS

In this section, we presented and discussed the results obtained. We identified 8,231 papers, of which 79 paper were read. The Kappa's value for the analysis was 0.612 (Substantial agreement). All disagreements were resolved by discussion that included the researchers, before proceeding to the next stage. The implementation results of the procedures are shown in Table II.

TABLE II. PRIMARY SELECTION

Sources	Total	Non-papers	Duplicated	Excluded	Included
IEEE	6,631	501	18	6,082	30
Compendex	33	1	1	25	6
Springer	292	4	0	282	3
ScienceDirect	293	6	0	285	3
Scopus	982	6	0	914	37
Total	**8,231**	**518**	**19**	**7,633**	**79**

For each source, the same search string was used, adjusting it to their restrictions. We found 8,231 papers distributed in 6,631 papers in IEEE (80.6%), 33 papers in Compendex (0.4%), 292 papers in Springer (3.5%), 293 papers in ScienceDirect (3.6%) and 982 papers in Scopus (11.9%). The greatest number of papers has been obtained in IEEE, with 501 papers of Table of Contents, Index, and Standards (7.5%), 18 duplicate papers (0.2%), and 6,082 irrelevant papers (91,3%). The result was only 30 relevant papers to our study (0.9%). In IEEE repository, we filtered the search by publication (Conference Publications, Journals, Magazines, and Standards), publication year, and topics. The options selection, according to the inclusion criteria, was done manually and the results obtained by the applying of the filters were combined.

In Compendex, we conducted the search in the Expert Search option. The results were 1 paper of Table of Contents (3%), 1 repeated paper (3%), 25 irrelevant papers (63.7%), and

6 relevant papers (30.3%). We used filters for Year (publication less than 2004), subject area (value Computer Science value), Document Type (values Conference Paper and Articles), Source Type (values Conference Proceedings and Journals), and Language (value English). In Springer, the search engine returned 4 repeated papers (0.6%), 282 irrelevant papers (93.1%), and 6 relevant papers (6.3%). The search was performed using the Advanced Search option, entering the search string in the with at least one of the words and where the title contains fields. Furthermore, it has cleared the Include preview-only content item to remove papers in-press. The filters used were Content Type (value Article), Discipline (value Computer Science), and Language (value English).

In ScienceDirect, we used the Expert Search option by selecting the Journals option. The Articles in press field was cleared. The filters used were Sources (value All Journals), Subject (value Computer Science), Limit by document type (values Articles, Review Article, and Short Survey), and Topic (values software, computer science, information system). The search engine returned 6 papers of Table of Contents and Index (2.0%), 285 irrelevant papers (96%), and two relevant papers (2.0%). In Scopus, the search engine returned 6 repeated papers (0.6%), 914 irrelevant papers (93.1%), and 37 relevant papers (6.3%). The filters used were Year (publications less than 2004); Subject Area (value Computer Science), Document Type (values Conference Paper and Articles), Source Type (values Conference Proceedings and Journals), and Language (value English).

The selected papers (79 papers - 1.6%) as primary studies are presented in Table III. In this table, # (reference to paper - P + sequential number), Year, Title, Source, and Authors of the papers are presented in chronological order (by year).

TABLE III. PAPERS SELECTED

#	Year	Title	Source	Authors
P1	2004	A Comparison of Cohesion Metrics for Object-Oriented Systems	Scopus	Etzkorn, L. H. Gholston, S. E. Fortune, J. L. Stein, C. E. Utley, D. Farrington, P. A. Cox, G. W.
P2	2004	A Methodology for Constructing Maintainability Model of Object-Oriented Design	IEEE	Kiewkanya, M. Jindasawat, N. Muenchaisri, P.
P3	2004	An Exploratory Study of Object-Oriented Software Component Size Determinants and the Application of Regression Tree Forecasting Models	ScienceDirect	Pendharkar, P. C.
P4	2004	Predicting Class Testability Using Object-Oriented Metrics	IEEE	Bruntink, M. van Deursen, A.
P5	2005	An Empirical Analysis of Object-Oriented Metrics for Java Technologies	IEEE	Farooq, A. Braungarten, R. Dumke, R. R.
P6	2005	An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite	Scopus	Succi, G. Pedrycz, W. Djokic, S. Zuliani, P. Russo, B.
P7	2005	Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction	Scopus	Gyimothy, T. Ferenc, R. Siket, I.
P8	2005	Measuring the Impact of Friends on the Internal Attributes of Software Systems	IEEE	English, M. Buckley, J. Cahill, T. Lynch, K.
P9	2006	A Complexity Metrics Set for Large-Scale Object-Oriented Software Systems	IEEE	Ma, Y. He, K. Du, D. Liu, J. Yan, Y.
P10	2006	Analyzing the Software Quality Metrics for Object Oriented Technology	Scopus	Parthasarathy, S. Anbazhagan, N.
P11	2006	Building Scalable Failure-Proneness Models Using Complexity Metrics for Large Scale Software Systems	IEEE	Bhat, T. Nagappan, N.
P12	2006	Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults	Scopus	Zhou, Y. Leung, H.
P13	2006	Empirical Study of Object-Oriented Metrics	Scopus	Aggarwal, K. K. Singh, Y. Kaur, A. Malhotra, R.
P14	2006	Identification of Defect-Prone Classes in Telecommunication Software Systems Using Design Metrics	Scopus	Janes, A. Pedrycz, W. Russo, B. Stefanovic, M. Succi, G.
P15	2006	Predicting Fault-Prone Components in a Java Legacy System	Compendex	Arisholm, E. Briand, L. C.
P16	2006	The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design	Scopus	Counsell, S. Swift, S. Crampton, J.
P17	2007	A Large-Scale Empirical Comparison of Object-Oriented Cohesion Metrics	IEEE	Barker, R. Tempero, E.
P18	2007	A Probabilistic Approach to Predict Changes in Object-Oriented Software Systems	IEEE	Sharafat, A. R. Tahvildari, L.

TABLE III. PAPERS SELECTED (CONT.)

#	Year	Title	Source	Authors
P19	2007	An Empirical Study of Class Sizes for Large Java Systems	IEEE	Zhang, H. Tan, H. B. K.
P20	2007	An Empirical Study of Slice-Based Cohesion and Coupling Metrics	Scopus	Meyers, T. M. Binkley, D.
P21	2007	Assessing Software System Maintainability using Structural Measures and Expert Assessments	IEEE	Anda, B.
P22	2007	Assessment of Package Cohesion and Coupling Principles for Predicting the Quality of Object Oriented Design	IEEE	Atole, C. S. Kale, K. V.
P23	2007	Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods	Scopus	Pai, G. J. Bechta Dugan, J.
P24	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	Olague, H. M. Etzkorn, L. H. Gholston, S. Quattlebaum, S.
P25	2007	Metrics and Evolution in Open Source Software	IEEE	Lee, Y. Yang, J. Chang, K. H.
P26	2007	Object-Oriented Software Fault Prediction Using Neural Networks	Compendex	Kanmani, S. Uthariaraj, V. R. Sankaranarayanan, V. Thambidurai, P.
P27	2007	Predicting Object-Oriented Software Maintainability Using Multivariate Adaptive Regression Splines	Scopus	Zhou, Y. Leung, H.
P28	2008	An Empirical Validation of Object-Oriented Class Complexity Metrics and Their Ability to Predict Error-Prone Classes in Highly Iterative, or Agile, Software: A Case Study	Compendex	Olague, H. M. Etzkorn, L. H. Messimer, S. L. Delugach, H. S.
P29	2008	An Empirical Validation of Object-Oriented Design Metrics for Fault Prediction	Scopus	Xu, J. Ho, D. Capretz, L.
P30	2008	Predicting the Maintainability of Open Source Software Using Design Metrics	Springer	Zhou, Y. Xu, B.
P31	2009	*Dn*-Based Design Quality Comparison of Industrial Java Applications	IEEE	Roubtsov, S. Serebrenik, A. van den Brand, M.

TABLE III. PAPERS SELECTED (CONT.)

#	Year	Title	Source	Authors
P32	2009	Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness Models	Scopus	Singh, Y. Kaur, A. Malhotra, R.
P33	2009	Examining the Potentially Confounding Effect of Class Size on the Associations Between Object-Oriented Metrics and Change-Proneness	Scopus	Zhou, Y. Leung, H. Xu, B.
P34	2009	Fault Detection and Prediction in an Open-Source Software Project	Scopus	English, M. Exton, C. Rigon, I. Cleary, B.
P35	2009	Identifying Thresholds for Object-Oriented Software Metrics	Science	Ferreira, K. A. M. Bigonha, M. A. S. Bigonha, R. S. Mendes, L. F. O. Almeida, H. C.
P36	2009	Is Depth of Inheritance Tree a Good Cost Prediction for Branch Coverage Testing?	IEEE	Shaheen, M. R. du Bousquet, L.
P37	2009	Predicting Code Change by Using Static Metrics	IEEE	Mauczka, A. Grechenig, T. Bernhart, M.
P38	2009	Quantitative Evaluation of Software Quality Metrics in Open-Source Projects	IEEE	Barkmann, H. Lincke, R. Lowe, W.
P39	2009	Investigation of Domain Effects on Software	Scopus	Virani, S. Etzkorn, L. Gholston, S. Farrington, P. Utley, D. Fortune, J.
P40	2010	A UML Approximation of Three Chidamber-Kemerer Metrics and Their Ability to Predict Faulty Code Across Software Projects	IEEE	Cruz, A. E. C. Ochimizu, K
P41	2010	A Hybrid Set of Complexity Metrics for Large-Scale Object-Oriented Software Systems	Springer	Ma, Y. He, K. Li, B. Liu, J. Zhou, X.
P42	2010	A Quantitative Approach to Software Maintainability Prediction	IEEE	Ping, L.
P43	2010	A Replicated and Refined Empirical Study of the Use of Friends in C++ Software	Scopus	English, M. Buckley, J. Cahill, T.
P44	2010	A Study of the Relationships between Source Code Metrics and Attractiveness in Free Software Projects	IEEE	Meirelles, P. Santos, C. Miranda, J. Kon, F. Terceiro, A. Chavez, C.

TABLE III. PAPERS SELECTED (CONT.)

#	Year	Title	Source	Authors
P45	2010	An Object-Oriented High-Level Design-Based Class Cohesion Metric	Scopus	Al Dallal, J. Briand, L. C.
P46	2010	Assessing Traditional and New Metrics for Object-Oriented Systems	Scopus	Concas, G. Marchesi, M. Murgia, A. Pinna, S. Tonelli, R.
P47	2010	Can Complexity, Coupling, and Cohesion Metrics be Used as Early Indicators of Vulnerabilities?	Scopus	Chowdhury, I. Zulkernine, M.
P48	2010	Change Prediction in Object-Oriented Software Systems: A Probabilistic Approach	Scopus	Sharafat, A. R. Tahvildari, L.
P49	2010	Finding Software Metrics Threshold Values Using ROC Curves	Scopus	Shatnawi, R. Li, W. Swain, J. Newman, T.
P50	2010	Size, Inheritance, Change and Fault-Proneness in C# Software	Scopus	Gatrell, M. Counsell, S.
P51	2010	Towards Estimating Physical Properties of Embedded Systems using Software Quality Metrics	IEEE	Corrêa, U. B. Lamb, L. Carro, L. Brisolara, L. Mattos, J.
P52	2010	Validation of CK Metrics for Object Oriented Design Measurement	IEEE	Kulkarni, U. L. Kalshetty, Y. R. Arde, V. G.
P53	2011	An Analysis of SNA Metrics on the Java Qualitas Corpus	Scopus	Tonelli, R. Concas, G. Marchesi, M. Murgia, A.
P54	2011	An Empirical Analysis of Lack of Cohesion Metrics for Predicting Testability of Classes	Scopus	Badri, L. Badri, M. Touré, F.
P55	2011	An Empirical Study on Object-Oriented Metrics and Software Evolution in Order to Reduce Testing Costs by Predicting Change-Prone Classes	IEEE	Eski, S. Buzluca, F.
P56	2011	An Empirical Verification of Software Artifacts Using Software Metrics	Scopus	Shatnawi, R. Alzubi, A.
P57	2011	Analysis of Quality of Object Oriented Systems Using Object Oriented Metrics	IEEE	Kayarvizhy, N. Kanmani, S.
P58	2011	Design Evolution Metrics for Defect Prediction in Object Oriented Systems	Springer	Kpodjedo, S. Ricca, F. Galinier, P. Guéhéneuc, Y. G. Antoniol, G.

TABLE III. PAPERS SELECTED (CONT.)

#	Year	Title	Source	Authors
P59	2011	Impact of Attribute Selection on Defect Proneness Prediction in OO Software	Scopus	Mishra, B. Shukla, K. K.
P60	2011	Improving the Applicability of Object-Oriented Class Cohesion Metrics	Scopus	Al Dallal, J.
P61	2011	Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics	IEEE	Al Dallal, J.
P62	2011	Transitive-Based Object-Oriented Lack-of-Cohesion Metric	Science	Al Dallal, J.
P63	2011	Using a Class Abstraction Technique to Predict Faults in OO Classes: A Case Study Through Six Releases of the Eclipse JDT	Scopus	Babich, D. Clarke, P. J. Power, J. F. Kibria, B. M. G.
P64	2011	Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities	Scopus	Chowdhury, I. Zulkernine, M.
P65	2011	Using Source Code Metrics to Predict Change-Prone Java Interfaces	IEEE	Romano, D. Pinzger, M.
P66	2012	A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes	Compendex	Al Dallal, J. Briand, L. C.
P67	2012	An Investigation of Design Level Class Cohesion Metrics	Scopus	Kaur, K. Singh, H.
P68	2012	Empirical Investigation of Fault Prediction Capability of Object Oriented Metrics of Open Source Software	IEEE	Singh, P. Verma, S.
P69	2012	Evaluating the Correlation Between Software Defect and Design Coupling Metrics	IEEE	Abuasad, A. Alsmadi, I. M.
P70	2012	Fault Prediction and the Discriminative Powers of Connectivity-Based Object-Oriented Class Cohesion Metrics	Scopus	Al Dallal, J.
P71	2012	Investigating Object-Oriented Design Metrics to Predict Fault-Proneness of Software Modules	IEEE	Rathore, S. S. Gupta, A.
P72	2012	The Ability of Object-Oriented Metrics to Predict Change-Proneness: A Meta-Analysis	Scopus	Lu, H. Zhou, Y. Xu, B. Leung, H. Chen, L.

TABLE III. PAPERS SELECTED (CONT.)

#	Year	Title	Source	Authors
P73	2012	The Impact of Accounting for Special Methods in the Measurement of Object-Oriented Class Cohesion on Refactoring and Fault Prediction Activities	Scopus	Al Dallal, J.
P74	2012	Validating the Effectiveness of Object-Oriented Metrics over Multiple Releases for Predicting Fault Proneness	IEEE	Rathore, S. S. Gupta, A.
P75	2013	A Fuzzy Classifier Approach to Estimating Software Quality	Scopus	Pizzi, N. J.
P76	2013	An Empirical Study of Source Level Complexity	IEEE	Liu Xiao
P77	2013	Investigation of Relationship Between Object-Oriented Metrics and Change Proneness	Compendex	Malhotra, R. Khanna, M.
P78	2013	Object-Oriented Class Maintainability Prediction Using Internal Quality Attributes	Compendex	Al Dallal, J.
P79	2013	Statistical Comparison of Modelling Methods for Software Maintainability Prediction	Scopus	Kaur, A. Kaur, K.

A. Quantitative Analysis

Quantitative analysis of the results of the two elaborate research questions is discussed in this section. The first question is:

```
Q1: What are the main metrics
(traditional and object-oriented)
investigated/used to evaluate the
software internal quality?
```

Papers were selected in which traditional and OO software metrics were used to evaluate internal quality of software. We identified 265 metrics. Top-15 most cited metrics in 10 or more papers are presented in Table IV. Those metrics are associated with five different attributes (Cohesion - 5 metrics, Complexity - 2 metrics, Coupling - 3 metrics, Inheritance - 2 metrics, and Size - 3 metrics). The most cited metric is Lack of Cohesion of Methods (LCOM) (40 citations).

Despite the large number of identified metrics, many are considered variations of well-established metrics, for example, CBO_IUB and CBO_U, variations of CBO [P78]. Other metrics were mentioned in only one study, because it is characterized a specific quality aspect, such as, IUC - used to quantify the interface type external class cohesion in Java [P44].

TABLE IV. MAIN METRICS USED TO EVALUATE INTERNAL QUALITY OF SOFTWARE

#	Metrics	Number of Citations	Attributes
1	Lack of Cohesion of Methods (LCOM)	40	Cohesion
2	Depth of Inheritance Tree (DIT)	37	Inheritance
3	Response for Class (RFC)	33	Coupling
4	Coupling Between Objects (CBO)	32	Coupling
5	Number of Children (NOC)	30	Inheritance
6	Weight Methods per Class (WMC)	28	Complexity
7	Lines of Code (LOC)	20	Size
8	Number of Methods (NOM)	18	Size
9	Cyclomatic Complexity (CC)	13	Complexity
10	Number of Attributes (NOA)	11	Size
11	Lack of Cohesion of Methods (LCOM2)	10	Cohesion
12	Lack of Cohesion of Methods (LCOM4)	10	Cohesion
13	Loose Class Cohesion (LCC)	10	Cohesion
14	Tight Class Cohesion (TCC)	10	Cohesion
15	Fan-out (FOUT)	10	Coupling

Regarding the product quality model in ISO/IEC 25010 [9], those metrics were used to evaluate the internal quality of software in the Maintainability, Functional Suitability, Security, Usability, and Reliability characteristics. In 76 papers, the selected metrics were used to verify whether they are good quality indicators in relation to the Maintainability characteristic (Table V). Regarding the Security characteristic, in two papers [P47] [P64], the authors discussed if software metrics are able to identify vulnerabilities in software. The Functional Suitability characteristic has been addressed in papers seeking to identify fault-prone classes [P75] [P76] and propensity to change [P40] [P45].

TABLE V. ISO/IEC 25010 QUALITY CHARACTERISTICS ADRESSED IN THE PAPER SELECTED

#	Characteristics	Number of Citations	References
1	Maintainability	77	[P1]-[P34], [P36]-[P62], [P64]-[P79]
2	Functional Suitability	4	[P40], [P45], [P75], [P76]
3	Security	2	[P47], [P64]
4	Usability	1	[P44]
5	Reliability	1	[P59]

Regarding the Maintainability characteristic, the metrics presented in Table IV were defined or used to

- Estimate the fault-prone OO classes, change propensity, and errors propensity;
- Reduce testing costs and degree of connectivity between classes;
- Identify classes that violate design principles;
- Predict maintainability in classes;
- Evolve open-source projects;
- Analyze variability of metrics and their impact on software in different domains.

In general, to evaluate the occurrence of such events, the selected studies used statistical techniques/models. These studies and the context of the use of these techniques are covered in:

Q2: What are the main statistical techniques/models used by researchers at the software engineering area to evaluate the software internal quality?

The selected papers have different methodologies and objectives. We identified 40 different techniques listed in Table VI. Many studies use one or more statistical techniques to characterize internal quality of software. The choice of the technique may be dependent on the research objectives.

TABLE VI. MAIN STATISTICAL TECHNIQUES/MODELS USED TO EVALUATE THE INTERNAL QUALITY OF SOFTWARE

#	Statistical Techniques/Models	Number of Citations
1	Descriptive Analysis	42
2	Correlation Tests	39
3	Logistic Regression	36
4	Artificial Neural Networks	13
5	Principal Component Analysis	7
6	Probability	3
7	Collinearity Analysis	3
8	Nonparametric Tests	3
9	Statistical Inference	3
10	Meta-Analysis	2

The most widely used technique was the Descriptive Analysis (42 citations), whose aim is to describe and summarize information of a population as mean, maximum, minimum, variance, and standard deviation. Then, Correlation Tests were used in 39 papers. The purpose of using these tests was to determine the relationship among software metrics, for example, those values obtained in complexity metrics are related to the values of size metrics (more lines and methods, it can be more complex). The most common type of used test was the Spearman test (16 papers), followed by Pearson test (14 papers), and Kendall test (two papers). Five studies did not specify the type of correlation test applied in their samples. The third technique most used was the Logistic Regression (univariate and multivariate). The context is to create fault prediction and models of changes in software. In some studies, only models based on univariate regression (model for each metric considered as variable) are proposed. Failure prediction and models of changes prediction are also proposed using Artificial Neural Networks with machine learning algorithms. Principal Component Analysis (PCA) was mentioned in seven studies, aimed at reducing the applicability of metrics and identifying dimensions in which these metrics together can explain software characteristics. In some studies, PCA was used to identify the most relevant metrics for the research purpose, and then, the correlation between those metrics were assessed using correlation tests mentioned above [P1] [P11] [P15] [P26] [P28] [P54].

Finally, some studies use probability tests [P11] [P18] [P35], collinearity analysis [P6] [P24] [P63], non-parametric tests such as the Wilcoxon test [P27] [P36] [P47], inference statistical techniques, as chi-square test [P10], fuzzy [P29] [P75], and meta-analysis [P72]. Other techniques such as discriminant analysis [P4], ANOVA [P39], ROC analysis [P48], MARS [P27], Confounding Effect [P33], Hidden Markov Model [P57], and Bayesian models [P23], were addressed only once.

B. Qualitative Analysis

The results suggest that there are no obvious choices for which metrics and what techniques the researchers should use. Statistical methods are chosen according to the research objectives; therefore, to show, to predict or to optimize are obtained by different methods. The analyzing of the 79 papers showed different results regarding the best metrics and the statistical results. We can note that some papers use existing databases metrics, as the NASA database [P27] [P29] and PROMISE database [P71]. There are those that set up their own databases, collecting the value of the metrics on the selected systems and do not becoming available in the scientific media.

Concerning the metrics, CK suite was the most used and cited as the results obtained in the quantitative analysis (Q1). Several papers link these metrics; because they to be pioneers, their use is justified as good predictors of OO code quality. In addition, one paper [P40] proposes the use of metrics in the evaluation of the software quality using UML models and compare the evaluation to the value of the metrics collected in the source code.

In relation to statistical techniques/models, most of papers used descriptive analysis to characterize software (e.g., CBO and LCOM) with high and low average values, respectively, may indicate a software with high coupling and low cohesion [P78]. However, descriptive analysis provides simple summaries about the sample and the observations made; nothing can be concluded about the relationship among the variables in study. With multivariate analysis, inferences can be made about the relationship among different variables, including dependence (multiple regression, discriminant analysis, and multivariate analysis of variance) and interdependence techniques (correlation tests, covariance, clustering, factor analysis, and correspondence analysis).

Among the models proposed in the literature, there are initiatives that contribute to identify dimensions or to classify internal attributes and reliability, and identifying metrics that predict quality characteristics [P21] [P57]. However, they do not cover metrics for complexity, inheritance, polymorphism, encapsulation, coupling, cohesion, and size. Also, there are studies that focus in prediction models and what metrics are good predictors of quality [P2] [P4] [P7] [P8] [P11] [P12] [P14] [P15] [P16] [P22] [P23] [P24] [P25] [P26] [P27] [P28] [P30] [P31] [P32] [P33] [P34] [P36] [P37] [P38] [P49] [P50] [P51] [P54] [P55] [P58] [P59] [P60] [P61] [P62] [P63] [P70] [P71] [P72] [P77] [P78] [P79]. In two studies [P18] [P65], whose focuses is on identifying software metrics as indicators of

vulnerabilities, merits of modeling are not reached. Therefore, it is not possible to know whether in the context of many variables, the metrics chosen by the authors are relevant in determining quality of software. Besides, there was not papers whose main objective was to evaluate similarity among software based on internal quality, i.e., a quality model based on cluster analysis.

The most used programming languages were Java (55%) and C ++ (21%). Software systems developed in others languages, such as, C (7%), C# (3%), Python (2%), PHP (1%), Perl (1%), and Ada (3%) also served as sample for research. Seven studies have disclosed language of the systems used. Some papers justify the preference for Software developed in Java due to its popularity in OO [P5] [P45] [P73].

Is worth mentioning that 6 papers had as sample proprietary software: i) [P17] - 12 software developed in Java whose company and its location could not be revealed; ii) [P21] - 4 software of Norwegian company; iii) [P11] - 2 versions of Windows; iv) [P31] - 2 financial software of an Irish company; v) [P14] - 5 software of an American company; and vi) [P4] - 1 software developed by Software Improvement Group (SIG). Other papers used open-source software. The option of using open-source software can be explained by the ease of obtaining them in online repositories. Its use has increased significantly in recent decades to the point of becoming influential in the global. In one study [P44], user satisfaction and quality of open-source software are achieved because of the community cooperation of users and developers to report failures, fix bugs, and add features of quick and organized way.

Another point to note is the variability of the population studied. Approximately, 17 papers considered that the software domain is a key part for the generalization of the results. In six studies [P1] [P26] [P35] [P43] [P45] [P71], the authors stated that the value of the metrics can vary in different domains for having different or alternative design. Those data suggest an open field for studies, especially empirical to:

- Evaluate the internal quality of software considering software domains;
- Evaluate internal quality of software considering the Maintainability characteristic, seeing the extent of the metrics, their properties, and relevance;
- Evaluate the internal quality in other characteristics defined in ISO/IEC 25010;
- Map and synthesize these features in models to determine if metrics are able to predict and characterize the quality of OO software. The currently proposed models are not generalizable and do not cover all metrics and attributes;
- Measure and estimate reference values for software domains, considering different metrics;
- Check similar features among software in different domains.

Potential threats to validity of this study are the ability to rule out publications about the issue, in cases that papers have no title, abstract, and well-defined keywords aligned to the topic under study.

C. Others Results

In this section, we presented some statistics related to the 79 papers that used 2,550 references. Top-10 papers with more citations, between 69 and 503 citations, are presented in Table VII. For example, the paper [P7] was cited 503 times by the 79 papers. Top-10 most cited papers are presented in Table VIII. The paper A Metrics Suite for Object Oriented Design is the most referenced paper by the 79 papers (68 citations). This result is because the paper validates the first suite of object-oriented metrics (CK suite), which also has the largest number of metrics used in studies. Top-10 papers that have higher number of self-citations are listed in Table IX. The self-citation is a reference to an author does in his own studies published as a paper in the same journal or other that often publishes. In this research, the author: Jehad Al Dallal has papers with more self-citations. We found 7 of 10 papers with such characteristic (higher proportion of self-citations).

TABLE VII. PAPERS WITH LARGEST NUMBER OF CITATIONS

#	Title	Journal/Conference	Number of Citations
[P7]	Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction	IEEE Transactions on Software Engineering	503
[P12]	Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults	IEEE Transactions on Software Engineering	200
[P28]	An Empirical Validation of Object-Oriented Class Complexity Metrics and Their Ability to Predict Error-Prone Classes in Highly Iterative, or Agile, Software: A Case Study	Journal of Software Maintenance and Evolution	162
[P15]	Predicting Fault-Prone Components in a Java Legacy System	5th ACM-IEEE International Symposium on Empirical Software Engineering	123
[P16]	The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design	ACM Transactions on Software Engineering and Methodology	102
[P23]	Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods	IEEE Transactions on Software Engineering	97
[P4]	Predicting Class Testability Using Object-Oriented Metrics	4th IEEE International Workshop on Source Code Analysis and Manipulation	84
[P27]	Predicting Object-Oriented Software Maintainability Using Multivariate Adaptive Regression Splines	Journal of Systems and Software	83

TABLE VII. PAPERS WITH LARGEST NUMBER OF CITATIONS (CONT.)

#	Title	Journal/Conference	Number of Citations
[P24]	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	78
[P13]	Empirical Study of Object-Oriented Metrics	Journal of Object Technology	69

TABLE VIII. MOST CITED PAPERS

#	Title	Author	Number of Citations	Year
1	A Metrics Suite for Object Oriented Design	Chidamber, S. R. Kemerer, C. F.	68	1994
2	A Validation of Object-Oriented Design Metrics as Quality Indicators	Basili, V. R. Briand, L. C. Melo, W. L.	33	1996
3	Object-Oriented Metrics that Predict Maintainability	Li, W. Henry, S.	30	1993
4	Exploring the Relationship Between Design Measures and Software Quality in Object-Oriented Systems	Briand, L. C. Wüst, J. Daly, J. W. Porter, V.	27	2000
5	Object-Oriented Metrics: Measures of Complexity	Sellers, B. H.	23	1996
6	A Comprehensive Empirical Validation of Product Measures for Object-Oriented Systems	Briand, L. C. Daly, J. W. Porter, V. Wüst, J.	20	1998
7	Object-Oriented Software Metrics	Lorenz, M. Kidd, J.	20	1994
8	Measuring Coupling and Cohesion in Object Oriented System	Hitz, M. Montazeri, B.	20	1995
9	Towards a Metrics Suite for Object-Oriented Design	Chidamber, S. R. Kemerer, C. F.	19	1991
10	Cohesion and Reuse in an Object-Oriented System	Bieman, J. M. Kang, B. K.	18	1995

TABLE IX. PAPERS WITH MOST SELF-CITATIONS

#	Title	Number of Citations	Number of Self-Citation
[P45]	An Object-Oriented High-Level Design-Based Class Cohesion Metric	62	14
[P62]	Transitive-Based Object-Oriented Lack-of-Cohesion Metric	31	12
[P78]	Object-Oriented Class Maintainability Prediction Using Internal Quality Attributes	92	11
[P66]	A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes	65	11
[P60]	Improving the Applicability of Object-Oriented Class Cohesion Metrics	59	10
[P73]	The Impact of Accounting for Special Methods in the Measurement of Object-Oriented Class Cohesion on Refactoring and Fault Prediction Activities	56	8
[P70]	Fault Prediction and the Discriminative Powers of Connectivity-Based Object-Oriented Class Cohesion Metrics	55	7

TABLE IX. PAPERS WITH MOST SELF- CITATIONS (CONT.)

#	Title	Number of Citations	Number of Self-Citation
[P64]	Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities	75	5
[P41]	A Hybrid Set of Complexity Metrics for Large-Scale Object-Oriented Software Systems	65	5
[P75]	A Fuzzy Classifier Approach to Estimating Software Quality	60	5

Comparing the results listed in Table VIII with the results listed in Table VII, there is no evidence of the author's papers in the mentioned list. This consideration leads to discussion about academic visibility, the bibliometric indexes and its consequences. Although the practice of self-citation is usual to inflate the Impact Factor of journals, the Thomson Reuters, responsible for examining the Journal Citation Records (JCR), takes the necessary measures in time to calculate the impact factor for a particular journal. As an example, the self-citation in [P45] represents approximately 22.6% of the references. In [P62], the self-citations percentage rises to 38.7%. Self-citation may be personal or interpersonal. What should be discussed is the circumstance of use of these two types [Vanz; Caregnato, 2003]. In the case of personal citation, there is an excessive incorporation by the author of old papers in his/hers most recent publications. This type of self-citation strategy could be considered such as a progression of a research theme of the same author or group, for example. In the second case, the interpersonal citation identifies a constant citation of papers of the same group by different authors of the same group.

It is worth mentioning that [P45] [P60] [P62] [P66] [P70] [P78] fits in the first case, since there is evidence that indicates progression of a research theme. The point to be questioned is whether there is an intentional exclusion of relevant productions to the topic under study by other authors. Even if self-citations, as a bibliographic element, is not intrinsically a reprobate attitude, results discussed herein are intended to show that there is relevant research on software quality.

V. LIMITATIONS AND THREATS TO VALIDITY

Some limitations and threats to validity can be identified in the study. For construct validity, the authors decided to have a more comprehensive search string as possible to capture papers considered most relevant to the research topic. However, even if the search string is properly derived from research questions, it is possible that significant results have been excluded in the combinations of the terms entered. Moreover, it is possible that the exclusion of papers related to the theme, for lack of information, for example, without title, abstract and well-defined keywords and aligned to the topic under study, has limited the number of papers that can accrue interesting information the research.

For internal validity, in order to minimize any threat, especially in the selection of primary studies and data extraction, the researchers involved in the SLR conducted their activities in parallel and any disagreement was discussed to

reach a consensus and properly measured by the Kappa coefficient. For external validity, although this study has considered only scientific papers as primary studies, their results can be generalized to both the scientific community and to the industry, since the metrics mentioned in the study are broad knowledge of software professionals.

VI. Conclusion

In this paper, the aim was to identify studies evaluating the internal quality of OO software, using statistical techniques. As result of the systematic mapping, 79 papers were selected. We identified 265 metrics used to evaluate software quality, but only 15 (the most cited) were considered to answer the first research question. On statistical techniques/models, we identified 40 techniques grouped in 10 most cited. The result suggests the existence of an extensive set of related metrics, but often can represent the same characteristic, i.e., derivations and/or similar adaptations. Additionally, the selected papers do not consider all existing properties (attributes) to characterize the software quality. Many of them focus on properties like complexity, cohesion, and size.

Among the study contributions, we can highlight the main metrics used to quantify the internal quality of OO software, a list of main statistical techniques used to evaluate software quality, and the main references to the topic in research. For future work, we suggest empirical studies to create models to evaluate the internal quality of software, using PLS-SEM. Besides, we encourage studies that summarizes metric's values into public databases to be used as reference in other empirical studies in software quality.

References

[1] Al Dallal, J. The Impact of Accounting for Special Methods in the Measurement of Object-Oriented Class Cohesion on Refactoring and Fault Prediction Activities. In: Journal of Systems and Software, 85(5), 1042-1057. 2012.

[2] Anda, B. Assessing Software System Maintainability using Structural Measures and Expert Assessments. In: International Conference on Software Maintenance. pp. 204-213. 2007.

[3] Bailey, J.; Budgen, D.; Turner, M.; Kitchenham, B.; Brereton, P.; Linkman, S. Evidence Relating to Object-Oriented Software Design: A Survey. In: International Symposium on Empirical Software Engineering and Measurement. pp. 482-484. 2007a.

[4] Bailey, J.; Zhang, C.; Budgen, D.; Charters, S.; Turner, M. Search Engine Overlaps: Do they agree or disagree? In: Second International Workshop on Realising Evidence-Based Software Engineering. pp.2-2. 2007b.

[5] Bansal, M.; Agrawal, C. P. Critical Analysis of Object Oriented Metrics in Software Development. In: International Conference Advanced Computing & Communication Technologies. pp. 97-201. 2014.

[6] Briand, L. C.; Wüst, J.; Daly, J. W.; Porter, V. D. Exploring the Relationship Between Design Measures and Software Quality in Object-Oriented Systems. In: Journal of Systems and Software, 51(3), 245-273. 2000.

[7] Chidamber, S. R.; Darcy, D. P.; Kemerer, C. F. Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis. In: IEEE Transactions on Software Engineering, 24(8), 629-639. 1998.

[8] Ferreira, K. A. M.; Bigonha, M. A. S.; Bigonha, R. S.; Mendes, L. F. O.; Almeida, H. C. Identifying Thresholds for Object-Oriented Software Metrics. In: Journal System Software, 85(2), 244-257. 2012.

[9] ISO/IEC 25010. Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models. 2011.

[10] Jabangwe, R.; Börstler, J.; Smite, D.; Wohlin, C. Empirical Evidence on the Link between Object-Oriented Measures and External Quality Attributes: A Systematic Literature Review. In: Empirical Software Engineering, 20(3), 640-693. 2015.

[11] Khan, Y. A.; Elish, M. O.; El-Attar, M. A Systematic Review on the Impact of CK Metrics on the Functional Correctness of Object-Oriented Classes. In: International Conference on Computational Science and Its Applications - Volume Part IV. pp. 258-273. 2012.

[12] Kitchenham, B. Procedures for Performing Systematic Reviews. Technical Report TR/SE-0401. Url: http://www.idi.ntnu.no/emner/empse/papers/kitchenham_2004.pdf. 2004.

[13] Landis, J. R.; Koch, G. G. The Measurement of Observer Agreement for Categorical Data. Biometrics, 33(1), 159-174. 1977.

[14] Orenyi, B. A.; Basri, S.; Jung, L. T. Object-Oriented Software Maintainability Measurement in the Past Decade. In: International Conference on Advanced Computer Science Applications and Technologies. pp. 257-262. 2012.

[15] Ping, L. A Quantitative Approach to Software Maintainability Prediction. In: International Forum on Information Technology and Applications. pp. 105-108. 2010.

[16] Riaz, M.; Mendes, E.; Tempero, E. A Systematic Review of Software Maintainability Prediction and Metrics. In: International Symposium on Empirical Software Engineering and Measurement. pp. 367-377. 2009.

[17] Souza, L. B. L. de; Maia, M. de A. Do Software Categories Impact Coupling Metrics? In: Working Conference on Mining Software Repositories. pp. 217-220. 2013.

[18] Tian, Y.; Chen, C.; Zhang, C. AODE for Source Code Metrics for Improved Software Maintainability. In: International Conference on Semantics, Knowledge and Grid. pp. 330-335. 2008.

[19] Vanz, S. A. S.; Caregnato, S. E. Estudos de Citação: Uma Ferramenta para Entender a Comunicação Científica em Questão. 9(2), 295-307. 2003.