External Quality Metrics for Object-Oriented Software: A Systematic Literature Review

Danilo Santos, Antônio Resende, Paulo Afonso Junior, Heitor Costa

danilobatista@posgrad.ufla.br, {tonio, pauloa.junior, heitor}@dcc.ufla.br

Department of Computer Science - Federal University of Lavras - Lavras - MG, Brazil

Abstract

Software quality metrics can be categorized into internal quality, external quality, and quality in use metrics. Although exist close relationship between internal and external software quality, there are not explicit evidences in literature that attributes and metrics of internal quality impact external quality. This is essential to know which metric to use according to the software characteristic that you want to improve. Hence, we carried out a systematic literature review for identifying this relationship. After analyzing 664 papers, 12 papers were studied in depth. As result, we found 65 metrics related to maintainability, usability, reliability, and quality characteristics as well as main attributes that impact external metrics (size, coupling, and cohesion). In follow, we filtered some metrics that have clear definitions, are appropriately related to the characteristic that purports to measure, and do not use subjective attributes in their computation. Therefore, these metrics are more robust and reliable to evaluate software characteristics. So, these metrics are better for use in practice by professionals working in the software market.

Keywords: External Quality, Internal Quality Metrics, Internal Quality Attributes, Software Quality.

1. Introduction

The demand for software quality is intense, mainly because of the increasing dependence of society on these software products [1, 2]. Hence, quality has been brought to the center of the development process because it is no longer seen as a competitive advantage factor, but an essential condition for software development organizations to produce software more competitive, with low costs and deadlines [2]. Although quality is critical for these organizations to obtain billing and membership of users in the market, the concept of quality is difficult to define, describe, understand, and measure [3]. Software quality is a multidimensional concept, has levels of abstraction and can be conceptualized in broad or specific way.

In the software context, quality can be defined as the degree to which the characteristics of one software or service satisfy explicit and implicit needs of stakeholders, adding value to the product or service [4]. For example, software has quality if it complies with the requirements established/specified by the owner, designer and client, such as, esthetics, functionality, design done in time, and estimated budget cost of the life cycle, operability, and maintainability [5].

One way to evaluate the software quality is using software metrics. These metrics can be categorized into three groups [4]: i) **internal quality metrics** relating to software measurement of the static attribute related to its architecture, *e.g.*, number of source lines and the coupling level; ii) **external quality metrics** derived from the evaluation of software behavior, *e.g.*, number of defects found in a test and duration of a maintenance task; and iii) **quality in use metrics** referring to how the software meets the users' needs in a specific context, *e.g.*, efficiency and satisfaction.

There is a close relationship between internal and external quality metrics, relationship highlighted in ISO/IEC 25010 [4]. In this standard, the internal quality attributes are specified in order to achieve external quality, recommending use internal metrics¹ with a strong relationship with external metricsc. Other works dedicated efforts to verify this relationship, such as:

- Code metrics are strongly related to the maintainability quality characteristic in object oriented software [6];
- Internal metrics measure complexity, cohesion, coupling, and size and have more consistency with some quality characteristics, such as, reliability and maintainability [7];
- Cohesion and coupling are predictors of Maintainability quality characteristic [8].

¹ In the remainder of text, the words "internal metrics" and "external metrics" will be used instead of the term "internal quality metrics" and "external quality metrics", respectively.

Although ISO/IEC 25010 displays attributes and internal metrics related to external metrics, these are not feasible to be used in practice as they do not have a clear measurement procedure, easy to understand [9]. An example is *Message Clarity*, whose value is obtained by dividing the number of understood messages by the number of implemented messages. However, in this metric, messages to be considered are not defined, whom must understand the messages, and if the messages should be evaluated in a time interval. In addition, many metrics presented in this standard are not based on static attributes, which may affect their value. For example, *Message Clarity* accounting for the understanding of messages, this metric is dependent on the evaluator and his knowledge about software analysis.

However, there is not a single review covering all the external metrics proposed in the literature, nor what are the attributes and the internal metrics that impact on the external metrics, and how this relationship is structured. With these uncertainties as a motivating factor, this work presents a Systematic Literature Review (SLR) to identify: i) what are the external software metrics and its characteristics?; and ii) what are the internal attributes that impact these characteristics?

The remainder of this paper is organized as follows. Goal, driving process of SLR, details used in the conduct of SLR, quantitative and qualitative analyzes of the results are discussed in Section 2. Limitations and threats to validity are pointed out in Section 3. Conclusions, contributions and suggestions for future work are presented in Section 4.

2. SYSTEMATIC LITERATURE REVIEW

The Systematic Literature Review (SLR) technique facilitates the research, selection, analysis, and organization of work (scientific papers) as the steps and the criteria laid down are well defined [10]. In addition, SLR facilitates obtaining and evaluating of evidences regarding a specific subject [11]. This makes possible to know the state of the art related to the subject matter in a practical way and limiting the theoretical foundation, avoiding fruitless approaches [12]. SLR is based on four phases (Fig. 1), which have iterations among them [10, 11, 13, 14]: i) **Planning**, reason for performing the SLR; ii) **Execution**, research is executed in sources identified in the previous phase, selecting relevant papers to the topic researched; iii) **Result Analysis**, the collecting, organization of extracted data from selected papers, and relevant analysis to these data are executed; iv) **Packaging**, results of the previous phases are stored.

Figure 1 : SLR Process

2.1. Planning

The purpose of this SLR is to identify object oriented metrics to evaluate the external quality of object oriented software and to relate which internal attributes affect those metrics. The identification of these metrics is restricted to object oriented technology to be widely used in software development in academia and industry; so, the result has significant contribution. Two research questions (RQ) are designed to achieve the set goal:

RQ1: What are the metrics responsible for evaluating the external quality of object oriented software?

RQ2: What are the internal attributes affected by these external metrics?

The way to carry out the search of the studies was based on Web search engines, in the fields (*metadata* and full text) available for their search engines. To be selected, the search source should provide an online search engine with options for: i) advanced search with the use of keywords; ii) filtering the results by publication year and area and/or type of publication; iii) search base must provide export query results in *BibTex* or *EndNote* format; and iv) to present invariance in search result when the same set of keywords is used. Hence, we chose six data sources, scientific papers repositories (Table 1). To perform the search, we used the following search string, built from keywords, their synonyms, and restrictions of search machines:

("external metrics" OR "external metric" OR "external measure" OR "external measures" OR "external measuring"
OR "external measurement" OR "external measurements")

AND

("Functionality Suitability" OR "Performance Efficiency" Compatibility OR Usability OR Reliability OR Security OR Maintainability OR Portability)

Table 1: Repositories of Scientific Papers

Sources	Links
IEEE Xplore	http://ieeexplore.ieee.org
Science Direct	http://www.sciencedirect.com/
Ei Compendex	http://www.engineeringvillage2.org
Scopus	http://www.scopus.com
Springer	http://www.springer.com/
ACM	http://dl.acm.org/

This string does not contain keywords to filter the internal attributes (RQ2). This decision is grounded in refinement tests of search string, where a more rigorous filter by entering keywords for internal attributes do not return studies to answer RQ1. Therefore, in this decision, the goal was to get as many studies that relate external metrics with software attributes. Hence, such studies can be analyzed to prove if they contain or not reports of impacts between internal attributes and metrics. This decision prevents interference in the outcome of RQ1 and RP2 allows respond appropriately.

The inclusion criteria were considered complete papers published in Journals or Conferences in the Computer Science area (application area of the results). Also, we considered papers in electronic format (without restricted access to their content) and content related to external quality metrics for object oriented software. Exclusion criteria were considered papers that had incomplete, duplicate texts, with restricted access to their content, and works that are not articles (*i.e.*, rules and table of contents).

To ensure the quality of obtained papers, widespread search bases were used. To ensure the impartiality of these results, four researchers (PA, PB, PC, and PD) performed the study selection employing the following procedure:

- PA, PB, and PC have established and improved the search *string* used in the SLR. Subsequently, this string was perfected by PA, PB, PC, and PD and was approved by PD, experienced researcher in the field;
- PA performed the search string in the selected sources and documented the results using JabRef software [15]. PA found and excluded studies not considered papers (*i.e.*, rules and table of contents), initial search;
- The studies found were individually evaluated by PA, PB, and PC, as the inclusion and exclusion criteria. This evaluation was carried out by reading title, abstract, and keywords (primary selection). Studies whose evaluations have caused doubts about the inclusion/exclusion were included. Finally, studies have been documented in a list of included/excluded papers with justifications;
- There was the intersection among the studies selected by PA, PB and PC, and these studies were documented (Intersect 1). In the event of any disagreement about the inclusion/exclusion of studies, PA, PB, and PC discussed and resolved. In continuing disagreement, the study was included. The studies excluded were documented in a list of excluded papers with justifications;
- PD assessed the work resulting from the search string considering title, abstract, and keywords. Subsequently, PD performed the intersection between the studies selected by him and the Intersection 1. PA, PB and PC disagreements were resolved;
- Finally, PA found and eliminated duplicated papers (title, authors, and summary that were equals) and incomplete, with restricted access to their content. Subsequently, the full text reading of the papers (secondary selection) was initiated by eliminating non-relevant items, which do not have appropriate content to answer the research questions, getting the resulting studies of this SLR.

2.2. Execution

The SLR was performed in April 2015, and the results obtained are shown in Table 2. Note that the same search string was used in the repositories of scientific papers and restrictions on research are detailed below. The total amount of obtained studies was 664, distributed in the repositories of scientific papers as follows:

- IEEE Xplore returned 128 studies, selecting 11 studies (8.6%) in the primary selection. The secondary selection, six non-relevant studies (54.5%) were found, getting to the final five primary studies (45.5%);
- Science Direct returned 201 studies, selecting only one study (0.5%) in the primary selection. In secondary selection, this study was considered non-relevant. Therefore, all studies of this base were discarded;

- Ei Compendex returned 47 studies, selecting only one study (2.1%) in the primary selection. In secondary selection, this study was considered incomplete. Therefore, all studies of this base were discarded;
- Scopus returned 125 studies, selecting three studies (2.4%) in the primary selection. In secondary selection, all these studies were selected as primary studies (100%);
- Springer returned 121 studies, none selected study in the primary selection and consequently none selected study in the secondary selection;
- ACM returned 42 studies, six studies (14.3%) were selected in the primary selection. The secondary selection found one study non-relevant (16.7%), two duplicated studies (33.3%) and one incomplete study (16.7%), resulting in the selection of two studies in the secondary selection.

Table 2: Results of the Systematic Literature Review

Sources	Home Search	Primary Selection	Secondary Selection			Results of Primary Studies	Included
			Non-relevant	Duplicated	Incomplete		
IEEE Xplore	128	11	6	0	0	5	-
Science Direct	201	1	1	0	0	0	-
Ei Compendex	47	1	0	0	1	0	2
Scopus	125	3	0	0	0	3	-
Springer	121	0	0	0	0	0	-
ACM	42	6	1	2	1	2	-
Total	664	22	7	2	2	10	2

At final, we selected 10 studies (1.5%) of the 664 studies obtained by RSL, where five of these studies (50%) are from the IEEE, three studies (30%) are from the Scopus, and two studies (20%) are from the ACM (Table 3). This significant reduction studies between the initial search and the primary selection is due to the large number of studies that do not adequately to supply the inclusion criteria; mainly for not having evidence in title, keywords, and abstract relationship between internal and external metrics. After performing an initial analysis of these studies, we verified the existence of two works used as a basis for writing two primary studies. Hence, in Table 3, these works were included (E1.1 and E8.1) related to the primary studies E1 and E8, respectively.

Table 3: Selected Studies for Systematic Literature Review

#	Year	Title	Base
E1	2004	N. Nagappan. "Toward the Software Testing and Reliability Early Warning Metric Suite". In: International Conference on Software Engineering, pp.60-63 2004.	IEEE
E1.1	2003	M. Vouk, N. Nagappan and L. Williams. "Towards the Metric Suite for Early Software Reliability Assessment". In: International Symposium on Software Reliability Engineering, pp. 238-239, 2003.	IEEE
E2	2007	I. Heitlager, T. Kuipers, and J. Visser. "A Practical Model for Measuring Maintainability - A Preliminary Report". In: International Conference on the Quality of Information and Communications Technology, pp. 30-39, 2007.	IEEE
E3	2008	D. Stavrinoudis and MK Xenos. "Comparing Internal and External Software Quality Measurements". In: Joint Conference on Knowledge-Based, pp. 115-124, 2008.	Scopus
E4	2008	M. Bombardieri and FA Fontana. "The Specialization of the Square Quality Model for the Evaluation of the Software." In: IEEE / ACM International Automated Software Engineering, pp. 110-113, 2008.	IEEE
E5	2008	D. Tamir, OV Komogortsev and CJ Muell. "An Effort and Time Based Measure of Usability". In: International Workshop on Software Quality, pp. 47-52, 2008.	ACM
E6	2008	K. Toshihiro. "Usability Evaluation Based on International Standards for Software Quality Evaluation." In NEC Technical Journal, pp. 27-32, 2008.	Scopus
E7	2009	N. Barbosa and K. Hiram. "Assessment of Software Maintainability Evolution Using C & K Metrics". In: Latin America Transactions, IEEE, vol. 11, no. 5, pp. 1232-1237, 2013.	IEEE
E8	2013	J. Scott, S. Smith and F. Castor. "Towards the Catalog of Object-Oriented Software Maintainability Metrics". In: International Workshop on Emerging Trends in Software Metrics, pp. 84-87, 2013.	IEEE
E8.1	2012	J. Scott, E. Barreiros, A. Almeida, F. Lima, A. Alencar, G. Lima, S. Smith and F. Castor. "Aspect-Oriented Software Maintenance Metrics: A Systematic Mapping Study." In: International Conference on Evaluation & Assessment in Software Engineering, pp. 253-263, 2012.	IEEE
E9	2014	J. Walters and N. Grumman. "Improving Software Metrics through while Providing Support Cradle to Grave". In: CrossTalk, pp. 26-30, 2014.	Scopus
E10	2014	KM Oliveira, S. Lepreux, Kolski C. and A. Seffah. "Predictive Usability Evaluation: Aligning HCI and Software Engineering Practices". In: International Conference on l'Interaction Homme-Machine, pp. 177-182, 2014.	ACM

2.3. Quantitative Analysis of the Results

Looking at the years of publication of the primary studies (Fig. 2), we can note that the relationship between internal and external metrics is studied for at least 12 years. In this figure, this relationship was most discussed in 2008 in four studies.

Figure 2 - Primary Studies of the Distribution by Publication Year

For existing quote between primary studies (Table 4), we can note that there is little relationship among them. This fact is because the studies returned by the SLR, although they are interrelated, for dealing with external metrics, they deal with external metrics for different software quality characteristics, i.e., each study is a characteristic of quality and a set of specific metrics.

Table 4 : Papers More Cited "Inside" Primary Studies

Primary Study Cites	Primary Study Cited
E 1	E 1.1
E 8	E 8.1
E 9	E 1

Checking the key quotes in common among the 12 studies, we can note that the ISO/IEC 9126 and ISO/IEC 25010 standards are widely referenced (Table 5). We omitted papers with less than four quotes. In Table 6, the most referenced authors of the primary studies are presented. The name of the cited author and the number of citations to the author are presented in this table. Authors with less than two references to his/her name were omitted. Michalis Xenos is the author most referenced with three quotes.

Table 5 : Papers More Cited "Outside" of the Primary Studies

Title	Studies Primary that References	Number
ISO / IEC 9126 [16] & ISO / IEC 25010 [4]	E2 / E3 / E4 / E5 / E6 / E7 / E9 / E10	8
The Complexity Measure [17]	E2 / E3 / E4 / E7 / E9	5
Software Metrics: A Rigorous and Practical Approach [18]	E 1 / E2 / E3 / E7	4

Table 6 : Authors More Cited "Inside" of the Primary Studies

Author	Number of Citations
Michalis Xenos	3
Dimitris Stavrinooudis	2
Fernando Castor	2
Juliana Saraiva	2
Laurie Williams	2
MladenVouk	2
NachiappanNagappan	2
SérgioSoares	2

Similarly, in Table 7, we presented authors most often cited out of the primary studies, excluding the authors who wrote the primary studies. They are related authors who obtained more than five quotes. S. R. Chidamber and

C. F. Kemerer are the most referenced authors in the primary studies. This is due to the studies published jointly and widely disseminated by these authors, when referring to the software metrics.

Table 7 : Authors More Cited "Outside" of Primary Studies

Author	Number of Citations
S. R. Chidamber	8
C. F. Kemerer	8
V. Basili	6
L. Briand	6
T. J. McCabe	6
W. Melo	6
M. J. Shepperd	6
A. Abran	5
N. E. Fenton	5
N. F. Schneidewind	5

Observing Fig. 3, we can note that the quality characteristic most addressed is maintainability; reliability and usability are in equal proportions. Although there are studies that address more than one characteristic, they did not consider the Functional Adequacy, Efficiency Performance, Compatibility, Security, and Portability quality characteristics.

Figure 3 - Quality Characteristics Most Addressed

Evaluating the distribution of primary studies in the types of publications, we can notice that most of the studies was published in conferences (Fig. 4). This supposedly indicates that the authors are looking for a quick feedback of the papers presented, as well as most appropriate environments to exhibit their work, answer questions, and receive suggestions to improve them.

Figure 4 – Quantity of Published Studies by Type of Publication

2.4. Qualitative Analysis of the Results

In this section, the research questions developed for the SLR are answered. In addition, discussions are made on the primary studies.

RQ1: What are the metrics responsible for evaluating the external quality of object oriented software?

The answer to this research question is presented for each primary study. However, a compilation of software metrics found, the characteristics and sub-characteristics that these metrics measure and the internal attributes that impact is shown in Table 8.

E1 - Toward a Software Testing and Reliability Early Warning Metric Suite

In this paper, based on a previous study (E1.1) by the same author, one suite of metrics is proposed to assess the reliability of software. So, E1.1 was added to the results of the SLR:

- **R1.** It indicates whether there are few test cases for large amounts of code. Its value is the ratio between the amount of test cases and the amount of code lines;
- **R2.** It indicates the number of test cases is suitable for the quantity of requirements. Its value is the ratio between the amount of test cases and the quantity of requirements;
- **R3.** It checks the correctness of R1, indicating whether there are cases of extensive use, covering lots of lines of code. Its value is the ratio of the number of tested lines of code and the amount of code lines;
- **R4.** It is used to "control" the R1 and R2. So, if there are few test cases and each has lots of calls to the software code, the developer will not be penalized for having wrong. Its value is the ratio of the number of statements and the number of lines of code;
- **C.** It indicates the accuracy of R3, since it measures the average number of lines of code executed by the code testing. Its value is the ratio of the amount of code lines and the number of test cases.

The metrics presented in this study can be measured directly on attributes obtained from the project, like test cases. In addition, for evaluating the cases of software testing, such metrics are clearly related to the software reliability.

E2 - A Practical Model for Measuring Maintainability - A Preliminary Report

In this paper, authors proposed a model to evaluate the software maintainability based on their properties. Furthermore, some software metrics are related to measuring these properties:

- **Lines of Code.** It measures the amount of lines of code in the software. In this study, this metric do not consider blank lines and lines with comments and can be obtained directly on the source code. Furthermore, this metric is related to maintainability, because the larger is a software, more costly is your maintenance;
- **Man Years Via Backfiring Function Points.** It measures the volume of one software regardless of technology and language. To determine its value, the amount of lines of code is recorded and one table is used to convert the amount of lines of code to function points. This metric can be obtained directly on attributes not dependent on the environment, source code, and conversion table. Furthermore, it relates to maintainability, because the larger software product volume more costly becomes their maintenance;
- **Cyclomatic Complexity per Unit.** It determines the complexity of a specific unit by reason software between the number of lines of code that fail and the number of lines of code. This metric can be obtained directly on software source code. Moreover, this metric is related to the maintainability, since the greater is complexity much time and resources are expended to make the maintenance;
- **Duplicated Blocks Over 6 Lines.** It determines the percentage of lines of code duplicated in a block. Its value is the ratio of the number of duplicated code lines and the number of lines of code. This metric can be obtained directly on the source code. Its relationship with the maintainability is proven because the higher the number of duplicated blocks more laborious becomes your maintenance;
- **Lines of Code per Unit.** It measures the individual size of each unit of code. Its value is the number of lines of code in the analyzed unit, except blank lines and lines with comments. This metric can be obtained directly on the source code. Its relationship with the software maintainability is confirmed because the higher the number of lines in a unit more hard becomes the maintenance action;
- **Unit Test Coverage.** It evaluates the coverage of a unit test. Its value is obtained using the Clover tool². Hence, by having its value measured indirectly through a tool, we do not know the adequacy of this measure to assess the maintainability and how subjective are the analyzed attributes;

² http://www.qbssoftware.com/products/Clover/overview/_prodclover

- **Number of Assert Statements.** It measures the quality of unit testing, verifying that this test performs checks proportionate to their coverage. For there is no reason to have a unit test that call many methods, which increases its coverage, because this test will be a unit test as well as tests that call many methods, but do not test their behavior and end up not testing anything. The value of this metric is the number of valid statements in a unit. Therefore, it is checked if there is only one unit test or a unit test invokes many methods, but does not evaluate. Facts that increase the unit test coverage, but do not test anything. This metric is assessed based on attributes taken directly from the project and refers to software maintainability because the higher the quality of testing, better will be the software quality and better will be its maintainability.

E3 - Comparing Internal and External Software Quality Measurements

In this study, authors used a technique based on the weighted classification of customer opinion to assess external characteristics of software. This technique is based on the technical QWCO_s (*Qualifications Weighted Customer Opinion with Safeguards*) referred to enter a question to assess the correctness of answers to a questionnaire. Finally, we obtain the following formula for calculating the external extent (External Measurements):

$$EM = QWCO_s = \frac{\sum_{i=1}^n (O_i * E_i * \frac{S_i}{S_t})}{\sum_{i=1}^n (E_i * \frac{S_i}{S_t})}$$

where O_i is the point of normalization of the i customer opinion, E_i is the classification by the customer i , S_i is the number of existing security issues in the questionnaire, and S_t is the number of these issues that the client answered correctly. So, S_i/S_t reveals the reliability rate of the questions answered by the customer. Multiplying this rate by the customer qualification, we give the weight of the opinion of each customer. By applying O_i , the opinion provided by the customer is normalized. Therefore, this metric reveals accurately the views of customers in relation to a characteristic of the software.

E4 - The Specialization of the Square Quality Model for the Evaluation of the Software

In this study, authors presented metrics that could be implemented in relation to each quality sub-characteristic related to reliability and maintainability quality characteristics defined by ISO/IEC 9126.

Reliability

- Maturity

$$\frac{MTF}{RMTF} + \frac{PTC}{TC}$$

where MTF is average failures in the period, $RMTF$ is the reference time between failures in a period indicating the failure rate in a period, PTC is the number of approved tests cases, and TC is total quantity of test cases, revealing the correctness of test cases. Therefore, this metric shows how mature is the software based on the number of detected faults and the correctness of the test cases. This metric works on questions that can be obtained directly from the software, through the recording of failures and test cases. In assessing the failure rate and the correctness of the test cases, we can get an indication of how mature is the software and also how reliable it is;

- Fault Tolerance

$$1 - \frac{NC}{NF} + \frac{PFTC}{FTC}$$

where NC is the number of software errors in the period, NF is the number of failures in the period, $PFTC$ is the number of failures that passed the test cases, and FTC is total of failed test cases, indicating the failure rate for cases test. NF may indicate the failure rate that turned into errors, and by subtracting the value of 1, reverses this amount result in failure rate that did not become errors. Hence, this metric shows how tolerant software developed is resistant to flaws in those not turned into errors, then it may indicate the software reliability. This metric can be obtained directly from questions of the software by evaluating how the software is fault tolerant, considering the amount of faults that its value is related;

- Recoverability

$$\frac{TO}{TO + Tr} + \frac{PRTC}{RTC}$$

where TO is the operating time in the period, Tr is the recovery time in the period (time rate that the software is operating in a given period), $PRTC$ is the number of cases of recoverability tests approved by the user, and RTC the number of cases of impairment tests (rate of test cases approved). This metric indicates how recoverable is the software by the accuracy of test cases constructed to recover the software and the time that the software is operable; therefore, this metric is indicative of software reliability. Basing on operation time, recovery time and the number of test cases as soon, this metric can be directly obtained on attributes measured on source code;

- Compliance

$$\frac{CRCI}{RCI}$$

where $CRCI$ is the number of items compatible with the reliability and RCI is the number of items that have to comply with reliability rules. This metric assesses the reliability compliance; so, it is directly related to this characteristic. However, its value is obtained based on reliability rules, which are not defined in the description of this metric; its value becomes subjective to the user that uses it;

Maintainability

- Analyzability

$$\frac{RMTT}{MTT}$$

where $RMTT$ is the average time to resolve an error and MTT is the reference time to resolve an error. This metric reveals how easy was the analysis of the software to find and solve a reported error, because if a maintainer solves the errors that software in less time than allocated to that task, it is assumed that this software has good analyzability, just a good maintainability. Furthermore, this metric is based on questions that can be obtained directly from the software project;

- Changeability

$$\frac{\frac{CTNL}{H}}{\frac{CTML}{H}} + \frac{MTR}{RMTR}$$

where $\frac{CTNL}{H}$ is the number of lines added per hour, $\frac{CTML}{H}$ is the number of lines that undergo maintenance per hour (rate of lines added suffering maintenance per hour), MTR is average time to satisfy a request, and $RMTR$ is time referring to satisfy a request (rate of facility to meet the request relative to the average). This metric reveals how easy is to make changes to one software, then it is a clear indication of software maintainability. Furthermore, this metric is based on attributes that can be obtained directly on source code, a fact that makes it less subjective;

- Stability

$$\frac{\sum_{mods} \frac{FCB_{mod}}{FAC_{mod}}}{MM} + \frac{MMB}{M}$$

where FCB_{mod} and FAC_{mod} are the number of failures before and after making changes to the software, respectively, MM is the number of modified modules in a period, MMB is the number of modified modules for bug fixes, and M is the number of application modules (rate modules modified in the software). This metric indicates how the changes affect other software modules, generating new failures because of their interdependence, and how stable is the software to avoid side effects after receiving modifications. Therefore, this metric is a software maintainability indicator and it is not a subjective metric, to base its calculations on questions that can be obtained directly from the software project;

- Testability

$$\frac{\sum_{mods} \frac{\sum_{mods_tcs} ExtMcCABE_{tc}}{ExtMcCabe_{mod}}}{N}$$

where $\sum_{mods_tcs} ExtMcCABE_{tc}$ is the cyclomatic complexity of the test case, $ExtMcCabe_{mod}$ is the complexity of the module, and N is the number of application modules. This metric indicates the average complexity of the test cases of the software modules, indicating the software maintainability. Furthermore, this metric is based on attributes obtained directly from source code, fact that make it not subjective and not dependent on the environment in which it is inserted;

- Compliance

$$\frac{CDCI}{RCI}$$

where $CDCI$ is the number of items compatible with maintainability and RCI is the number of items that have to comply with the rules of maintainability. This metric indicates the ratio of items in accordance with set of maintainability rules and it is related to maintainability. However, this metric does not define what maintainability rules to be followed, a fact that makes subjective and impairs its use in practice.

E5 - An Effort and Time Based Measure of Usability

In this study, authors proposed external metrics to measure operability, learnability, and understandability, depending on usability effort. To determine this effort the user expends to perform an action, it is suggested the formula:

$$E(t) = \frac{1}{t - t_0} \int_{t_0}^t (w_1 * mc(t) + w_2 * mk(t) + w_3 * mic(t) + w_4 * p(t)) dt$$

where $mc(t)$, $mk(t)$, and $mic(t)$ are how many clicks the mouse, the number of keystroke and the number of *pixels* crossed in a time interval $(t - t_0)$, respectively. Furthermore, $p(t)$ is a penalty factor by switching between mouse and keyboard, or vice versa. With the result and the evaluation of a set of interfaces, we can define which one is best from the point of view of operability, learnability or understandability comparing the value of the effort of each one.

E6 - Usability Evaluation Based on International Standards for Software Quality Evaluation

In this study, authors held a discussion on the metrics provided in ISO/IEC 9126 concerning the Usability quality characteristic:

- **Understandable input and output.** It measures the comprehensiveness, making sure that the user properly understand what was required as software input and output. Its value is the ratio of the number of input and output items that the user understood and the number of input and output items available at interface. These metric is dependent on the user judgment that will decide whether an generated output was understandable; this metric becomes subjective and depends on your application context;
- **Learnability.** It measures how long the user takes to learn the software functions. Its value is the average time that user takes to properly learn the functions of this software. Therefore, this metric is not based on subjective attributes;
- **Operability.** It measures the proportion of times the user has applied the recovery operation correctly when faults occur. Its value is the ratio of the number of faults that the user has applied the recovery operation correctly and the number of tested faults. This metric is subjective and depends on your application context;
- **Attractiveness.** It measures the proportion of elements that can be customized in accordance with the preference of the user interface. Its value is the ratio of the number of elements that can be customized and the number of items that user wants to customize. This metric considers to its calculation data obtained directly from software design (items that can be customized); so, it is less subjective to the context in which the software is employed;

- **Usability Compliance.** It measures how much the software complies with the standards, guidelines, and guides related to usability. Its value is the ratio of the number of usability compliance items implemented during the test and the number of specified usability items. For considering good usability practices to determine compliance with this characteristic, this metric is subjective to the context in which it is used, as its value is influenced according to the good practice guide used.

E7 - Assessment of Software Maintainability Evolution Using C & K Metrics

In this study, Chidamber and Kemerer suite (CK) of metrics [19] was used to evaluate the software maintainability. For this, a statistical analysis was held to show "strong" relationship between this suite and maintainability. These metrics are listed and detailed in Table 8.

E8 - Towards a Catalog of Object-Oriented Software Maintainability Metrics

In this study, reference is made to a previous work of the same authors (E8.1), which presents a set of software metrics relating to maintainability. These metrics are listed and detailed in Table 8.

E9 - Improving Software Metrics through while Providing Support Cradle to Grave

In this study, CK metrics [19] and the metrics proposed by ISO/IEC 9126 are considered good indicators of software maintainability. These metrics are listed and detailed in Table 8.

E10 - Predictive Usability Evaluation: Alingning HCI and Software Engineering Practices

In this study, a set of metrics is presented to evaluate interface characteristics extracted from QUIM model (Quality in Use Integrated Measurement) to evaluate the usability characteristic. This model was proposed for specifying and identifying quality components considering different factors like metrics and data defined in Computer Interface and Software Engineering Models:

- **Function Understandability.** Its value is determined as the evaluation of Software Engineer in determining your proper understanding of an existing function in the software. This metric is subjective to the context in which it is used, because it depends directly of the knowledge of Software Engineer performing evaluation;
- **Longest Depth.** It measures the inheritance in a software, accounting for the number of nodes in the hierarchy from the root that is performed early task. This metric indicates what should be measured and is based in static attributes of software;
- **Visual Coherence.** It measures the visual consistency of the interface. Its value is the ratio of the number of primitive tasks unrelated to tasks at the same level in the interface and the number of primitive tasks defined in the interface. Although it is based on well, this metric does not specify how to establish relationships to classify levels of tasks, a fact that complicates understanding and calculation and makes the judgment subjective to the implementer;
- **Minimal Action.** It measures which is the shortest path, less effort, which can be made to perform a task. Its value is determined by the number of steps to access the desired primitive task. This metric is based on attributes that can be adequately measured independently of the environment, making it not subjective;
- **Input Validity Date.** It measures the number of data valid input in the software. Its value is the ratio of the number of sequence "information passing permission" plus the number of explicit tasks defined to validate the data and the number of decomposed tasks on at least a primitive task. This metric does not define what should be considered to make its calculation, leaving gaps for interpretation of the implementer that can cause mismatches in their calculation and biases in the conclusion obtained.

Table 8 : Results

#	Internal/External Metrics	Characteristic/Subcharacteristic	Internal Attribute	Studies
1	R1	Reliability	-	E1 / E1.1
2	R2	Reliability	-	E1 / E1.1
3	R3	Reliability	-	E1 / E1.1
4	R4	Reliability	-	E1 / E1.1
5	C	Reliability	-	E1 / E1.1
6	<i>Lines of Code</i>	Maintainability	Size	E2
7	<i>Man Years Via Backfiring Function Points</i>	Maintainability	Size	E2
8	<i>Cyclomatic Complexity per Unit</i>	Maintainability	Complexity	E2
9	<i>Duplicated Blocks Over 6 Lines</i>	Maintainability	Size	E2

10	<i>Lines of Code per Unit</i>	Maintainability	Size	E2
11	Unit test coverage	Maintainability	-	E2
12	<i>Number of Assert Statements</i>	Maintainability	-	E2
13	Maturity	Reliability	-	E4
14	Fault Tolerance	Reliability	-	E4
15	recoverability	Reliability	-	E4
16	Compliance	Reliability	-	E4
17	Analyzability	Maintainability	-	E4
18	changeability	Maintainability	-	E4
19	Stability	Maintainability	-	E4
20	testability	Maintainability	Complexity	E4
21	Compliance	Maintainability	-	E4
22	Understandable input and output	Usability	-	E6
23	Learnability	Usability	-	E6
24	operability	Usability	-	E6
25	Attractiveness	Usability	-	E6
26	Usability Compliance	Usability	-	E6
27	<i>Coupling Between Object Class</i>	Maintainability	Coupling	E7 / E8 / E8.1 / E9
28	<i>Number of Children of the Class</i>	Maintainability	Inheritance	E7 / E8 / E8.1 / E9
29	<i>Lack Cohesion in Methods</i>	Maintainability	Cohesion	E7 / E8 / E8.1 / E9
30	<i>Depth of Inheritance Tree</i>	Maintainability	Inheritance	E7 / E8 / E8.1 / E9
31	<i>Response for Class</i>	Maintainability	Inheritance	E7 / E8 / E8.1 / E9
32	<i>Weight Methods for Class</i>	Maintainability	Complexity	E7 / E8 / E8.1 / E9
33	<i>Lines of Code</i>	Maintainability	Size	E8 / E8.1
34	<i>Message Passing Coupling</i>	Maintainability	Coupling	E8 / E8.1
35	<i>Class Coupling</i>	Maintainability	Coupling	E8 / E8.1
36	<i>Data Abstraction Coupling</i>	Maintainability	Coupling	E8 / E8.1
37	<i>Tight Class Cohesion</i>	Maintainability	Cohesion	E8 / E8.1
38	<i>Lose Class Cohesion</i>	Maintainability	Cohesion	E8 / E8.1
39	<i>Lack Cohesion in Methods 3</i>	Maintainability	Cohesion	E8 / E8.1
40	<i>Number of Class</i>	Maintainability	Size	E8 / E8.1
41	<i>Lack Cohesion in Methods 2</i>	Maintainability	Cohesion	E8 / E8.1
42	<i>Lack Cohesion in Methods 1</i>	Maintainability	Cohesion	E8 / E8.1
43	<i>Number of Methods</i>	Maintainability	Size	E8 / E8.1
44	<i>Number of Attributes</i>	Maintainability	Size	E8 / E8.1
45	<i>Concern Diffusion Over Components</i>	Maintainability	-	E8 / E8.1
46	<i>Audit Trail Capability</i>	Maintainability / Analyzability	-	E9
47	<i>Diagnostic Function Support</i>	Maintainability / Analyzability	-	E9
48	Failure analysis capability	Maintainability / Analyzability	-	E9
49	<i>Failure Analysis Efficiency</i>	Maintainability / Analyzability	-	E9
50	<i>Status Monitoring Capability</i>	Maintainability / Analyzability	-	E9
51	<i>Change Cycle Efficiency</i>	Maintainability / Changeability	-	E9
52	<i>Change Implementation Elapse Time</i>	Maintainability / Changeability	-	E9
53	<i>modification Complexity</i>	Maintainability / Changeability	-	E9
54	<i>parameterized modifiability</i>	Maintainability / Changeability	-	E9
55	<i>Software Change Control Capability</i>	Maintainability / Changeability	-	E9
56	<i>Change Success Ration</i>	Maintainability / Stability	-	E9
57	<i>Modification Impact Localization</i>	Maintainability / Stability	-	E9
58	<i>Availability of Built-in Test Function</i>	Maintainability / Testability	-	E9
59	<i>Re-Test Efficiency</i>	Maintainability / Testability	-	E9
60	<i>Test Traceability</i>	Maintainability / Testability	-	E9
61	<i>Function understandability</i>	Usability	-	E10
62	<i>Longest Depth</i>	Usability	-	E10
63	<i>Visual Coherence</i>	Usability	-	E10
64	<i>minimal Action</i>	Usability	-	E10
65	<i>Input Validity Date</i>	Usability	-	E10

"-" Means that the primary study showed no such content.

RQ2: What are the internal attributes affected by these external metrics?

Among the primary studies, 41.7% (E2, E4, E7, E8 and E9 - 5 studies) made explicit in its content which internal attributes affect external metrics, presenting at least one metric (Table 8 - "Internal Attribute"). This table presents which characteristic or sub-characteristic is measured by the specified metric.

In conducting this analysis, we noted that internal and external metrics are related to external characteristics. This reinforces the existence of relationships between them, as the software characteristics can be determined based on attributes and internal metrics and can be measured by external metrics. External metrics are impacted by attributes and internal metrics. Analyzing which are the internal attributes that most affect the external characteristics, we obtained size, cohesion, and coupling (Fig. 5). The most impacted by internal attributes are Maintainability, Usability, and Reliability characteristics (Fig. 6).

Figure 5 - Internal Attributes Affecting External Metrics

Figure 6 - Quality Characteristics more Affected

A fact to be emphasized is even searching for external metrics in a comprehensive manner in the literature, metrics addressing the Functional Adequacy, Efficiency Performance, Compatibility, Security, and Portability quality characteristics were not identified. This does not mean that these metrics do not exist, but studies that address these characteristics can be used nomenclatures incompatible with ISO/IEC 25010 to address the issue. So, they were not detected by the search string used.

2.5. Discussion of Primary Studies

2.5.1. Discussion Among the Primary Studies

By comparing the results presented by the primary studies, we noted that only the study E2 employed efforts to introduce new external metrics to determine external characteristics. These metrics were based on internal software attributes. These attributes are more reliable for be independent of the environment in which the software runs. However, this primary study has not been evaluated, keeping the bad usual practice of proposing metrics and do not validate.

In other studies, E7, E8, E9 and E4 (for a single metric), the relationship among internal metrics widely used in the literature was evident, *e.g.*, CK suite and the software characteristics, in particular for these studies, Maintainability. E7, E8, and E9 showed convergence among metrics proposed. An interesting fact is these metrics belong to CK suite. However, only E7 evaluated the relationship, the other studies were based on the third-party statements.

E3 is intended to reinforce the idea that the domestic metrics relate to external characteristics. For this, this study was evaluated based on the correlation between internal and external metrics. The result showed that there is significant correlation between both. In addition, we found that good internal structure implies high maintainability, based on the opinion of developers.

Changing the focus to the external metrics that assess external attributes, E1 indicates that internal metrics are excellent indicators of external software quality. However, few metrics were evaluated to show that what they measure is really what they purport to measure. Another point is these metrics do not explicit as they relate to the external characteristics of software. This fact was also confirmed by E8 and E9. Hence, E1 proposed and evaluated five external metrics. However, this assessment is performed using software developed by university students. This

may endanger the evaluation of the work, as these software have been developed by professionals with possible low experience.

Still evaluating external metrics that measure external attributes, there are two ways: **i**) studies reporting external metrics presented in ISO/IEC 9126-2 [16] or ISO/IEC 25010 [4] (E4, E6, and E9); and **ii**) studies that criticize the metrics presented by these standards (E8 and E9). This criticism is due to the fact that the metrics presented in these standards have high cost of implementation and are difficult to apply in practice. The arguments used by studies contrary to the standards are strong and valid, but, as can be seen in this SLR, few authors who ventured to propose software metrics without subjective factors involved in its calculation.

Leaving aside these subjective factors, E5 proposed a different point of view of the studies discussed above. In this study, external metrics are evaluated in relation to the effort that is expended to carry out a task in software systems. The evaluation of the work is carried out by university students in a travel booking software. This can endanger the evaluation, because these students may not have consolidated enough knowledge to conduct this kind of task.

Focusing the discussion on the metrics presented by primary studies, we noted that they have different software metrics (Table 8), existing convergence among the metrics identified (column - "Studies"). However, some metrics can be divided into two groups:

- Metrics with the same name and different way to measure. Among the studies that show these metrics, there are E2, E8, and E8.1 for LOC metric. E2 considers LOC as the quantity of lines of code in the project, except for the blank lines and lines that have comments. E8 and E8.1 consider all design lines, including lines with comments and blank;
- Metrics with different name and same way to measure. This is the case of the Compliance Usability and Compliance Maintainability metrics. These metrics have the same formula and seek to evaluate how much the characteristic analyzed is in accordance with the rules set for this characteristic.

Analyzing the metrics obtained in this SLR, we noted that some of them tend to be used in groups, such as CK suite, in which the authors used metrics that compose it instead of only a few specific metrics. This suite contains metrics considered most relevant in this SLR, as are those with higher amounts of the articles referencing. Other relationships could not be established because some articles not properly detailed how the reported metrics are measured. Hence impossible to claim that the metrics are similar or not.

2.5.2. Discussion Among Software Characteristics and its Metrics

With the implementation of this SLR, we identified 65 software metrics aimed Maintainability, Reliability, and Usability characteristics. To bring to light the representation that such metrics have in relation to their respective characteristics, then we discussed how subjective are the basis of each metric identified and if this metric presents properly evidence to reflect the characteristic to which it refers. This discussion is a compilation of comments make on each metric in Section 2.4.

Maintainability characteristic refers to the ability of the software can be modified efficiently to meet the evolving, adaptive, and corrective needs [4]. In this SLR, we identified 46 metrics aimed at maintainability. Some of these metrics (6-12, 19, 20, and 27-45) are clearly defined and have their values directly measured about source code through issues such as lines of code, cyclomatic complexity, methods, and hierarchy tree. Therefore, its implementation in practice becomes feasible because such metrics can be easily implemented to measure the source code and provide a context-independent response in which they are employed.

Although they are clearly defined, some of these metrics (11 and 18) does not reach its goal of measuring maintainability, because it does not consider any factor defined in characteristic or have no apparent relationship with this characteristic. For example, changeability metric analyzes the number of added and modified lines and does not check whether these changes have generated failures or degraded software performance as expressed in ISO/IEC 25000.

Other metrics related to maintainability (17, 21 and 46-60) do not clearly define which attributes should be considered in their calculations and how their values can be obtained. For example, analyzability metric is based on the principle that the smaller the ratio of time spent to accomplish a task and the time allocated for its implementation, the greater the software analyzability. However, this concept is vague and context dependent, as it does not define what should be considered to estimate the time of a task. Therefore, the same software can be characterized with analyzable or not, according to the criteria used to estimate the execution time of a task. Although the metric indicates what question to be measured, this is subjective and dependent on external factors to the analyzed software. Moreover, it is only possible to have knowledge of analyzing parts of the software

which have been modified or received some changes. Hence, the analyzability will be measured relative to only one portion of the software and not the software as a whole.

In relation to usability, we identified 10 metrics. Usability characteristic refers to the ability of the software to be understood, learned, used, and attractive to the user under certain conditions [4]. Some of these metrics (23, 25, 62 and 64) have their set of values obtained directly about the software and have direct relationship with the software usability.

Other metrics aimed at Usability (22, 24, 26, 61, 63 and 65) have their values on subjective attributes or have not had their measurement so well specified in the studies analyzed. This fact makes it difficult to implement in practice, because according to the user and the environment in which the metric is used, different questions can be measured, resulting in different metrics and different conclusions. Hence, these metrics can lead to bias in the evaluation of software.

Regarding Reliability characteristics, we obtained 9 metrics. This characteristic defines the ability of a product, component or system to perform its function properly under certain conditions [4]. Analyzing the metrics concerning to Reliability, we found that most of these metrics (1-5 and 13-15), represents 88% of the total, are related to maintainability. Moreover, these metrics define the requirements to be measured to perform their calculations. Hence, only the #16 metric does not use questions aims to define their values because it used in its calculation the notion of items compatible with usability, but does not define them.

Based on the metrics obtained in the SLR and discussion presented above, a set of less subjective metrics can be filtered (Table 9). This table shows the metrics has its clear definitions, appropriately related to the characteristic that purports to measure and not using subjective questions in their calculations. Hence, the use of non-subjective metrics to assess the three discussed characteristics is facilitated.

Table 9: Indicated Software Metrics for Analyzing Software Characteristics

Software Characteristic	Identifier of Metrics
Maintainability	6, 7, 8, 9, 10, 12, 19, 20, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45
Usability	23, 25, 62, 64
Reliability	1, 2, 3, 4, 5, 13, 14, 15

2.5.3. New Questions

As result of reading and discussion of the results of 12 primary studies, four new questions arose and were answered based on the content extracted from these studies.

What motivated the authors to study the external metrics?

The motivating factor in most studies (E1, E2, E3, E4, E5, E8 and E9 - 7 studies) is convergent. This is because at least part of their motivations reported that existing metrics in the literature, proposed by a standard or other primary study, are subjective, uncertain, costly to be implemented in practice, incomprehensible - not indicate precisely what attributes should be modified to improve the value of the measure - and often have not been validated.

Are primary studies empirical or theoretical?

Among the primary studies, 33% of the studies are theoretical, studies that discuss, depth, and enhance the knowledge of a point of the search area. The rest of the studies obtained (67% of studies) is empirical studies which seek through experiments reach new knowledge grounded in other studies and maturation of authors' knowledge. This reveals that the authors are devoting efforts to find a solution of the gap existing in the relationship between internal and external metrics.

How do variation of attributes and internal metrics affect external metrics?

The primary studies inform that internal metrics or internal attributes generate impacts on the external and/or external characteristics measured and do not specify how the behavior (increase or decrease) in one affects the other. Therefore, this question cannot be answered by this SLR, getting open to be answered by future research.

Do papers suggest or use tools to measure the metrics presented?

Only two studies explicit in its content tools to measure the metrics presented (Table 10). Hence, 7 of 65 metrics (approximately 10%) had a related tool in the content of studies to assess. This fact is not an indication that there are not tools to assess other metrics, because there are multiple tools in the literature and on the market to measure the metrics, such as, CodePro AnalytiX³; Metrics⁴; JDepend⁵; CodeAnalyzer⁶, and DependencyFinder⁷. Therefore, the low percentage of tools for measuring can be justified by the fact that it was not the purpose of the analyzed studies highlight these tools on their content.

Table 10: Cited Tools to Measure the Metrics Obtained in SLR

Tool	Primary Study	Identifier of Metrics
Clover	E2	11
CKJM8	E7	27, 28, 29, 30, 31, 32

3. THREATS TO VALIDITY AND LIMITATIONS

Some limitations may affect the results of this SLR. One is the possibility of primary studies have not used consistent nomenclature with the defined by ISO/IEC 25010 [4] to refer to the external characteristics of software quality. Another limitation refers to studies that did not relate the metrics presented to an external characteristics of software quality. Hence, significant studies may not be found because of the different terms used in this defined standard, which were used in the search string of this SLR.

The search for studies written only in the English language can be characterized as another limitation, because work with relevant content written in other languages may not have been found. Similarly, books, thesis, and dissertations were excluded which could provide enriching content for research, featuring another limitation.

The set of used search sources can be another limitation, because other sources may present relevant content to the topic searched. The return of the bases used in this SLR characteristics is another limitation because it cannot ensure that they adequately address the search string used. Hence, there is no guarantee that were returned "all" primary studies contained in their collections and that they should really be listed according to the *string* used.

Regarding the threats to validity, one refers to the relationship between internal and external metrics with their software quality characteristics. Some studies did not address what quality characteristic each metric relates. For this reason, in some cases, we inferred the characteristic that the metric is related based on the intent and content of the primary study.

Biases in the evaluation of researchers during the selections of primary studies characterize another threat to validity, as these selections are subjective. However, the selection of the studies was conducted by four researchers (PA, PB, PC and PD) in isolation to minimize this threat. When there was different opinions among researchers PA, PB and PC, another (senior) researcher (PD) was consulted to resolve the impasse.

4. CONCLUSIONS

In this paper, we presented the results of conducting an SLR on external metrics of software. Initially, 664 studies were returned by SLR, with the primary selection only 22 studies were selected. However, some studies were discarded: 7 studies were non-relevant, 2 studies were duplicated between the bases, and 2 studies were incomplete. As a final result, we selected 10 primary studies. Subsequently, 2 studies were included for presenting content related to two selected primary studies, totaling 12 studies analyzed.

³ <https://marketplace.eclipse.org/content/codepro-analytix>

⁴ <http://metrics.sourceforge.net/>

⁵ <http://www.clarkware.com/software/JDepend.html>

⁶ <http://www.codeanalyzer.tecl.ws/>

⁷ <http://depfind.sourceforge.net/>

⁸ <http://www.spinellis.gr/sw/ckjm/>

The two research questions were **RQ1: What are the metrics responsible for evaluating the external quality of object oriented software?** and **RQ2: What are the internal attributes affected by these external metrics?** These questions were answered by the analysis of 12 primary studies.

As for **RQ1**, we found 65 software metrics (Table 8) and other ways to measure external characteristics of software to analyze the usability based on the efforts made in the execution of a task. These metrics are related mainly to Maintainability, Usability, and Reliability quality characteristics.

By analyzing which internal attributes generated impacts on those metrics, **RQ2**, we identified that size, cohesion, and coupling are attributes responsible for these impacts. Whereas the internal and external metrics are related to the same external characteristics [4] and based on the responses of research questions, to perceive the existence of the relationship between both. This is because software quality characteristics can be determined by attributes and internal metrics and measured by external metrics. Therefore, the external metrics also are impacted by attributes and internal metrics.

The measurement of quality characteristics based on attributes and internal metrics makes the metrics more accurate, reliable, and less costly, as these attributes and these metrics depend on the environment where the software runs. For these reasons, it is possible and advisable to determine external characteristics of software based on internal attributes.

Although only E2 determined external characteristics based on internal attributes, some primary studies (E1, E3, E7, E8 and E9 - 5 studies) indicate the existence of relationships between attributes and internal metrics and external metrics. Considering the number of empirical studies identified and the increased frequency of papers published in recent years, we can note that the academy is seeking to solve the gap of lack of jobs that relate attributes and internal metrics to external metrics.

The contribution of this paper lies precisely on this gap, as this SLR gathers knowledge about the relationship between attributes and internal metrics with external metrics. In addition, it also presented a set of metrics judged as more robust because it is composed of metrics that has a way of clearly defined measurement, informed in questions that can be obtained from source code or software design and showing full relationship with the characteristics that purports to measure. It is also presented tools used by primary studies to measure some of this metrics. These facts allow practitioners working in the market know what metrics to use, according to the quality characteristic that wish to measure. This measurement allows these professionals to work on improving the quality characteristic measured, which consequently leads to improved quality of the software.

As future work, a new SLR study could be conducted including primary studies written in other languages, using other sources of scientific papers and inclusion of thesis, dissertations, and books. In addition, it would be interesting to conduct some experiments aimed at using some of the metrics identified in this SLR to evaluate the external quality of real software.

REFERENCES

- [1] R. Pressman and B. Maxim. "Software Engineering: A Practitioner's Approach." McGraw-Hill. 976p. 2014.
- [2] S. H. Kan. "Metrics and Models in Software Quality Engineering". 2nd Edition. Addison-Wesley Longman Publishing Co., Inc., 560P. 2002.
- [3] B. A. Kitchenham and J.G. Walker. "A Quantitative Approach to Monitoring Software Development". In: Software Engineering Journal. v.4, no. 1. pp. 2-13. 1989.
- [4] ISO/IEC 25010. "Systems and Software Engineering - Systems and software Quality Requirements and Evaluation - System and Software Quality Models" 2011.
- [5] A. A. Bubshait. "Owner Involvement in Project Quality". In: International Journal of Project Management. v.12, no. 2, pp. 115-117. 1994.
- [6] W. Li and S. Henry. "Object-Oriented Metrics that Maintainability Predict". In: Journal of Systems and Software, vol. 23, no. 2, pp. 111-122, 1993.
- [7] R. Jabangwe, J. Börstler and D. Smite. "Empirical Evidence on the Link Between Object-Oriented Measures and External Quality Attributes: A Systematic Literature Review". In: Empirical Software Engineering, pp. 1-54, 2013
- [8] J. Al Dallal. "Object-Oriented Class Maintainability Prediction Using Internal Quality Attributes". In: information and Software Technology, vol. 55, no. 11, pp. 2028-2048, 2013.
- [9] D. Rachida and M. Bari. "An ISO 9126 based Quality Model for the e-Learning Systems." International Journal of Information and Education Technology. v.6, no. 5, 370p, 2016.
- [10] B. Kitchenham. "Procedures for Performing Systematic Reviews." Keele University Technical Report TR / SE-0401. Available: <<http://www.inf.ufsc.br/~awangenh/kitchenham.pdf>>. Accessed on: April / 2015.

- [11] J. Biolchini, P. G. Mian, Natali A. C. C. and G. H. Travassos. "Systematic Review in Software Engineering". UFRJ, Rio de Janeiro, pp. 1-31, 2005.
- [12] R. Justus. "A Guide to Writing the Dissertation Literature Review". In: Evaluation & Practical Assessment Research, p. 13, 2009.
- [13] G. H. Travassos and J. Biolchini. "Systematic Reviews Applied Software Engineering". In: Brazilian Symposium on Software Engineering. 2007
- [14] B. Kitchenham. "Guidelines for Performing Systematic Literature Review in Software Engineering". EBSE Technical Report, 2.3, Keele University, pp. 1-57, 2007.
- [15] JabRef [Online]. Available at: <. [Http://jabref.sourceforge.net/](http://jabref.sourceforge.net/)> Accessed on: April / 2-15.
- [16] ISO/IEC 9126-2. "Software Engineering - Product Quality - Part 2: External Metrics (ISO / IEC TR 9126-2)". Geneva: International Organization for Standardization and the International Electro technical Commission, 2003.
- [17] T. J. McCabe. The Complexity Measure. In: IEEE Transactions on Software Engineering. v.2, no. 4. pp. 308-320. 1976.
- [18] N. E. Fenton and S. L. Pfleeger. "Software Metrics: A Rigorous and Practical Approach". Course Technology. 656 p. 1998.
- [19] S. Chidamber and C.F. Kemerer. "Towards a Metrics Suite for Object-Oriented Design". In: Conference on Object-Oriented Programming ng Systems, Languages, and Applications, pp. 197-211, 1991.