PERSPECTIVASonline CIÊNCIAS EXATAS E ENGENHARIA

Revista Perspectivas Online: Exatas & Engenharia
June/2021, v.11, n.31, p.33-59
ISSN: 2236-885X (Online)
DOI: 10.25242/885X113120212284

LITERATURE REVIEW ON SOFTWARE METRICS AND A NEW PROPOSAL

Mara Regina dos Santos Barcelos[1], Carlos Francisco Simões Gomes[1], Adriana Manzolillo Sanseverino[1*] & Marcos dos Santos[2]

ABSTRACT

BARCELOS, MR.R.S.; GOMES, C.F.S.; SANSEVERINO, A.M.; SANTOS, M. Literature review on software metrics and a New Proposal. **Perspectivas Online: Exatas & Engenharias**, v. 11, n. 32, p. 33- 59, 2021.

The use of metrics is important in software development activities as they make it possible to check quality, identify failures and other benefits. The objective of this paper is to propose a new software metric based on a bibliometric study and a literature review on software metrics. The bibliometric research was carried out in the Scopus and Web of Science databases to identify the distribution of articles by year of publication, the main authors, affiliation, country, the most common languages, the types of documents, journals with more publications, areas of knowledge, and the keyword clusters. Twenty-three articles were subsequently selected for reading to compose the literature review. The results of the bibliometric research show that (i) there is no defined core of research; (ii) there is a fluctuation of the number of published articles; (iii) the predominant language is English, and the country with the highest index of publications is the United States; (iv) the main area of knowledge is computer science; (v) in relation to affiliation, Florida Atlantic University stands out; (vi) the journal with the largest number of publications is the Journal of Systems and Software. The literature review showed that many software metrics can be used for different purposes, but most of them are related to code, and none are related to acceptance. As such, a support metric for the software acceptance process is proposed to facilitate the delivery phase of the software product, providing security for the customer and cost savings for the developing company.

Keywords: software metrics; bibliometric, literature review; software quality.

[1]Universidade Federal Fluminense – Departamento de Engenharia de Produção - Rua Passo da Pátria, 156, São Domingos, Niterói, RJ, CEP: 24210-240, Brasil;
[2]Instituto Militar de Engenharia - Praça General Tibúrcio, 80, Praia Vermelha, Urca, Rio de Janeiro, RJ, CEP: 22290-270, Brasil.
(*) e-mail: adrianams@id.uff.br

Data de recebimento: 27/02/2021 Aceito para publicação: 21/04/2021 Publicado online: 22/06/2021

Revista Perspectivas Online: Exatas & Engenharia
June/2021, v.11, n.31, p.33-59
ISSN: 2236-885X (Online)
DOI: 10.25242/885X113220212284

REVISÃO DE LITERATURA SOBRE MÉTRICAS DE SOFTWARE E UMA NOVA PROPOSTA

Mara Regina dos Santos Barcelos[1], Carlos Francisco Simões Gomes[1], Adriana Manzolillo Sanseverino[1*] & Marcos dos Santos[2]

ABSTRACT

BARCELOS, MR.R.S.; GOMES, C.F.S.; SANSEVERINO, A.M.; SANTOS, M. Literature review on software metrics and a New Proposal. **Perspectivas Online: Exatas & Engenharias**, v. 11, n. 32, p. 33- 59, 2021.

O uso de métricas é importante nas atividades de desenvolvimento de software, pois permitem verificar a qualidade e identificação de falhas, entre outros benefícios. O objetivo deste artigo é propor uma nova métrica de software baseada em um estudo bibliométrico e uma revisão da literatura sobre métricas de software. A pesquisa bibliométrica foi realizada nas bases de dados Scopus e Web of Science para identificar a distribuição dos artigos por ano de publicação, principais autores, afiliação, país, idiomas mais comuns, tipos de documentos, periódicos com mais publicações, áreas do conhecimento e os grupos de palavras-chave. Posteriormente, 23 artigos foram selecionados para leitura a fim de compor a revisão da literatura. Os resultados da pesquisa bibliométrica mostram que (i) não há um núcleo de pesquisa definido; (ii) há flutuação do número de artigos publicados; (iii) o idioma predominante é o inglês e o país com maior índice de publicações são os Estados Unidos; (iv) a principal área de conhecimento é a informática; (v) em relação à afiliação, a Florida Atlantic University se destaca; (vi) o periódico com maior número de publicações é o Journal of Systems and Software. A revisão da literatura mostrou que muitas métricas de software podem ser usadas para diferentes propósitos, mas a maioria delas está relacionada ao código e nenhuma está relacionada à aceitação. Dessa forma, uma métrica de suporte ao processo de aceitação do software é proposta para facilitar a fase de entrega do produto de software, proporcionando segurança para o cliente e redução de custos para a empresa desenvolvedora.

Palavras-chave: métricas de software; bibliometria, revisão de literatura; qualidade de software.

[1]Universidade Federal Fluminense – Departamento de Engenharia de Produção - Rua Passo da Pátria, 156, São Domingos, Niterói, RJ, CEP: 24210-240, Brasil;
[2]Instituto Militar de Engenharia - Praça General Tibúrcio, 80, Praia Vermelha, Urca, Rio de Janeiro, RJ, CEP: 22290-270, Brasil.
(*) e-mail: adrianams@id.uff.br

Received: 27/02/2021 Accepted: 21/04/2021 Published online: 22/06/2021

1. INTRODUCTION

For Pressman and Maxim (2016), software consists of instructions that provide characteristics, functions and performance, in addition to having a data structure that enables the correct manipulation of information. For Wazlawick (2019), several concepts and practices should be used to produce software with quality and reliability.

Figure 1 illustrates the software features, including the methodology, the type of code, the type of software and the customers.

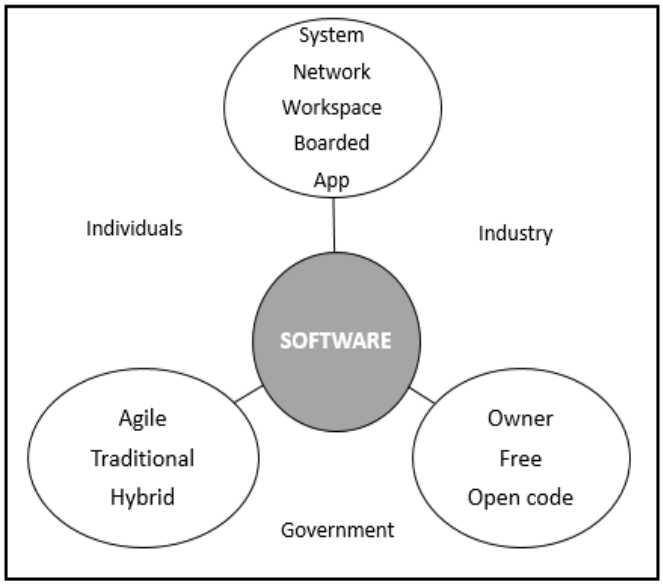

Figure 1. Software features.

The software development process involves several activities, regardless of the size and complexity of the software or the type of development adopted by the team. Pressman and Maxim (2016) describe a set of software development activities: (i) Communication; (ii) Planning; (iii) Modelling; (iv) Construction; (v) Delivery.

The demand for information technology (IT) resources increases every day, whether for business or entertainment purposes. The IT sector, which is made up of hardware, software and services, drives the national economy. According to Beranic and Hericko (2019), different approaches to derive software metric thresholds based on benchmark data are available in the existing literature. For Sheetal and Ravindranath (2018), the categories of software metrics have advantages and deficiencies.

Each year, the Brazilian Association of Software Companies (ABES) conducts research on the IT sector in partnership with IDC (International Data Corporation), providing information on investment, growth, software development and marketing companies, among others.

Table 1 shows the data from the surveys conducted in the period from 2013 to 2019. There is a fluctuation in the growth index of investments in the IT sector, with a decrease of

3.6% in 2016. However, the financial activity of the sector is considerable, reaching more than $70 billion in the years 2013 to 2015.

Table 1. Annual evolution of the IT sector. Source (ABES, 2020)

Year	Investment growth index in the IT sector (%)	Financial Activity (US$) (Software, Hardware, Services)	Growth Index of the Software Sector (%)	Number of software development and commercialization companies
2013	15.4	72.3 bi	13.5	8,302
2014	6.7	71.2 bi	12.8	9,308
2015	9.2	72.3 bi	30.2	10,140
2016	-3.6	48.0 bi	0.2	11,237
2017	4.5	47.6 bi	2.8	11,237
2018	9.8	47.7 bi	3.3	11,944
2019	10.5	44.3 bi	16.0	12,248

The global IT market in 2018 generated $2.234 billion, of which $1.220 billion in software and services (ABES, 2020).

The growth index of the software sector also fluctuates, reaching 30.2% in 2015. The number of companies focused on software development and commercialization is increasing, totalling 12,248 in 2019.

These facts justify the need to develop quality software products, as technological advances occur in increasingly shorter periods, requiring frequent updates of the software. According to Sommerville (2019) this need arose in the 1960s. At that time, software was slow and difficult to maintain and / or reuse.

In addition, the performance evaluation of academic production is particularly important for researchers, research organizations, research development agencies and government agencies (Zhang; Wu, 2020).

In this context, this paper presents a bibliometric study and a literature review on software metrics. The bibliometric study was useful to select an initial set of references on the topic and answer the following questions:

- How many papers are published by year?
- Which keywords are used more frequently? How do they relate?
- Which countries have published the most about the theme?
- Which areas of knowledge have published more papers about the theme?
- Which journals have published more papers about the theme?
- Which authors have published more papers, and how are they connected?
- What are the main institutions that publish papers on the subject?

This paper is organized as follows: section 2 clarifies the concepts; section 3 describes the research methodology; section 4 provides the results of the bibliometric study; section 5 presents the selected articles on software metrics; section 6 proposes a support metric for the software acceptance process; and section 7 highlights the conclusions.

2. CLARIFYING CONCEPTS

There is more than one definition for software quality. For De Pádua Paula Filho (2010), the quality of a software product refers to its degree of adherence to the established requirements. Likewise, Pressman and Maxim (2016) say that the quality of a software project is related to the degree of adherence to the functions and characteristics specified in the requirements. ISO 9126-1 (ABNT, 2003) defines the quality of the software as being the totality of the characteristics of an entity that allow it to meet the explicit and implicit requirements.

Quality is related to all software product development activities. According to Hassan et al. (2020) an automated extraction of usability requirements can be crucial during the review and validation phases, as it is difficult to determine them before real users use the software.

In Brazil, SOFTEX (Association for the Promotion of Excellence in Brazilian Software) has been promoting support actions aimed at improving software quality. In 2003, SOFTEX created the MPS-Br (Brazilian Software Improvement Program) to increase the competitiveness of companies, improving their processes. This program is divided into seven levels of maturity (A-G), and provides specific guidelines according to each level, which must be implemented for the company to be certified.

Da Silva et al. (2019) conducted a study to identify how companies use software to support quality management activities. The results indicate that, according to the companies participating in the study, the software in use meets three of the quality management principles established in ISO 9001: (i) approach to the process; (ii) improvement, (iii) evidence-based decision making.

Melluzzi Neto et al. (2018) carried out a survey to verify the degree of satisfaction of the senior management of ten software development companies in Maringá. The maturity models considered in the study were MPS-Br and CMMI. The results show that, according to the managers, the adoption of a maturity model has brought many benefits, such as standardization of processes and improvement in the planning capacity of the team.

Freitas e Monteiro (2017) used process simulation to identify the best service flow, from a help desk system to a professional council, using the Arena software. By these means, they obtained a useful simulation model that could assist in decision making.

Freitas, Albuquerque and Santos (2017) applied a unified study using criteria based on portfolio management and PMBOK management, and the MPS-Br Acquisition process for Services. Their goal was to verify the opinion of public institutions and suppliers on the use of guides, standards, and maturity models to support the acquisition and contracting of

information technology projects in Brazilian public administration. The results revealed that the majority of respondents do not use a service procurement protocol, but the greatest difficulty reported was the lack of a project manager to conduct the process.

In this context, quality management techniques were developed to provide significant improvements in software products. Metrics are one of these techniques.

The definition of a metric, analysis tool, methodology and technique must therefore include (Tempero and Ralph, 2018; Misirlis and Vlachopoulou, 2018): the entities to which it applies; mapping the characteristics of these entities to measure the number of symbols; the attribute that it purports to measure.

Both the quality and the measurement process in MPS-Br are at the F (managed) level. The objective of the measurement process is to collect, store, analyse and report the organization's data related to its projects, the products developed, and the processes implemented. The purpose of the quality assurance process is to ensure that the products and the execution of the processes follow the established plans, procedures, and standards (SOFTEX, 2016).

According to Fenton and Bieman (2014), the measurement process converts symbols and / or numbers into attributes of a real-world entity. The software measurement process can have different objectives: verifying the quality of the product; establishing the productivity of the team; formulating estimates, among others (Pressman and Maxim, 2016).

Java Metrics (CKJM) are the most frequently used metrics for measuring the design and quality of the software system. This methodology is an aggregation of: Logistic Regression (LOGR), Decision Tree (DT), Artificial Neural Network with Gradient Descent (AGD), ANN with Gradient Boosting (AGDX), ANN with Radbas Function (ARBF), Support Vector Machine with linear kernel (SLIN), SVM with polynomial kernel (SPOLY), SVM with RBF kernel (SRBF), Extreme Learning Machine with linear kernel (ELIN), ELM with polynomial kernel (EPOLY), ELM with RBF kernel (ERBF), Least Square SVM with linear kernel (LSLIN), LSSVM with polynomial kernel (LSPOLY), LSSVM with RBF kernel (LSRBF), and two different assembling techniques, such as Majority Voting Ensemble Methods (MVE) and Best Training Ensemble (BTE) (Tummalapalli et al., 2020).

Tripathi et al. (2020) identify descriptive statistics: response time, availability, throughput, successibility, reliability, latency, maintainability, modularity, reusability, testability, interoperability, conformity.

Sheetal and Ravindranath (2018) identify the following categories: process metrics (deployed to assess the costs of the software development process or the duration or attributes of the methodology); project metrics (deployed to evaluate the software project status, cost, number of employees and skill sets of the employees); and product metrics (deployed to evaluate the product at any phase of the development). Project metrics are also classified according to static metrics and dynamic metrics.

3. METHODOLOGY

The methodology includes: (i) a bibliometric study using the Scopus and Web of Science databases, since they are the most used in academic works (Zhu; Liu, 2020); (ii) a literature review on software metrics; and (iii) a proposal for a new metric.

(i) The bibliometric research was performed according to the Webibliomining model, suggested by Costa (2010) and Silva *et al*. (2015), to analyse the scientific production on the topic of software metrics, considering the following topics: Year of publication; Types of documents; Keyword clusters; Languages; Main authors; Network of authors; H index of authors; Affiliation; Countries; Knowledge areas; Journals.

Bibliometric methods provide a quantitative approach, aggregating a large amount of bibliographic data; they are an alternative to qualitative reviews (Vogel; Günttel, 2013). In this article, the bibliometric method provided an accurate selection of an initial set of references on software metrics based on a descriptive overview of the scientific production.

The CAPES Portal (www.capes.gov.br) provided the access to the databases in December 2020. The searches were carried out according to the following structures:

a) The Scopus database: The advanced search was used, initially the strategy was TITLE-ABS-KEY ("software metrics"), resulting in 3,668 documents. We decided to use a more specific strategy, considering the purpose of this research: TITLE ("software metrics"), resulting in 642 documents, of which 197 were articles.

b) The Web of Science database: The advanced search was used with the strategy: TI = ("software metrics"), resulting in 369 documents, of which 135 were articles.

c) Using the VOSviewer software: The VOSviewer software was used to create maps of the keyword clusters and the author network. This practice emphasizes the most common keywords and the connections between the authors on the topic under analysis.

(ii) Twenty-three articles were selected to compose the literature review on software metrics through the bibliometric study.

(iii) A new software metric was proposed.

4. RESULTS OF THE BIBLIOMETRIC STUDY

The results show a fluctuation (Figure 2). Publications started in 1978 in the Web of Science database and in 1979 in the Scopus database. The highest number of publications occurred in 2015 in the Web of Science database with 26 documents, and in 2011 in the Scopus database with 37 documents.

Since the beginning of publications in the Scopus database (1979), there was no period without publications; in the Web of Science database, however, there were no publications from 1979 to 1982.

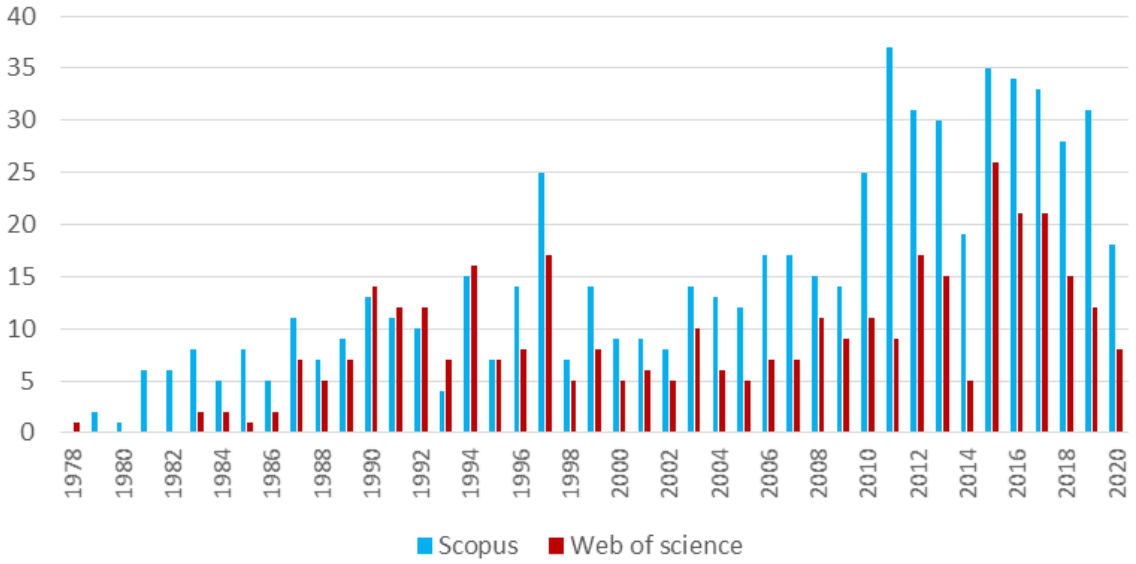

Figure 2: Publications between 1978 and 2020.

Regarding the types of documents, conference articles are predominant with 402 results in the Scopus database and 222 in the Web of science database (Figure 3).

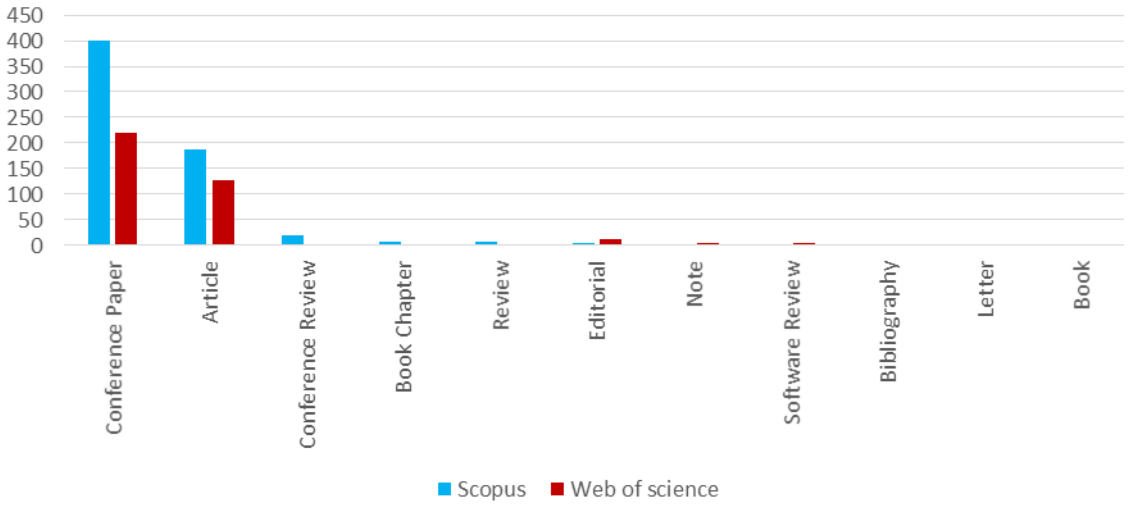

Figure 3: Types of documents.

The English language predominates with 632 results in the Scopus database and 365 results in the Web of science database (Figure 4).

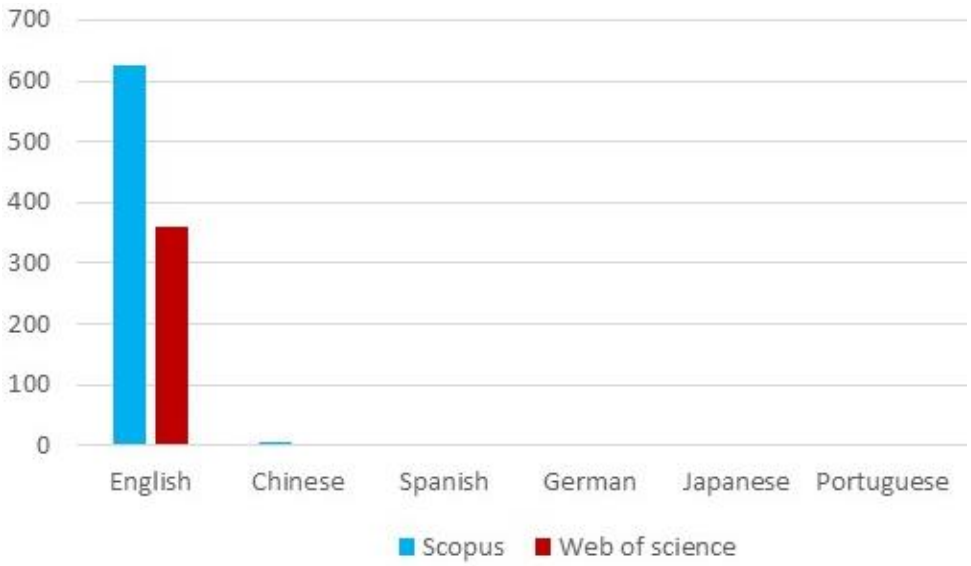

Figure 4: Languages.

Analysing the words can return information and knowledge about a certain subject (Ishikiriyama et al., 2015). The VOSviewer software was used to create the keyword clusters. We analysed 2,982 keywords, of which 207 met the minimum limit of 5 occurrences in publications. Figure 5 illustrates the intensity of occurrence of these keywords, Figure 6 shows the densities of these keywords, and Figure 7 illustrates the top keywords with more than 50 records. The keywords with the highest index of occurrence are software metrics with 410 occurrences.

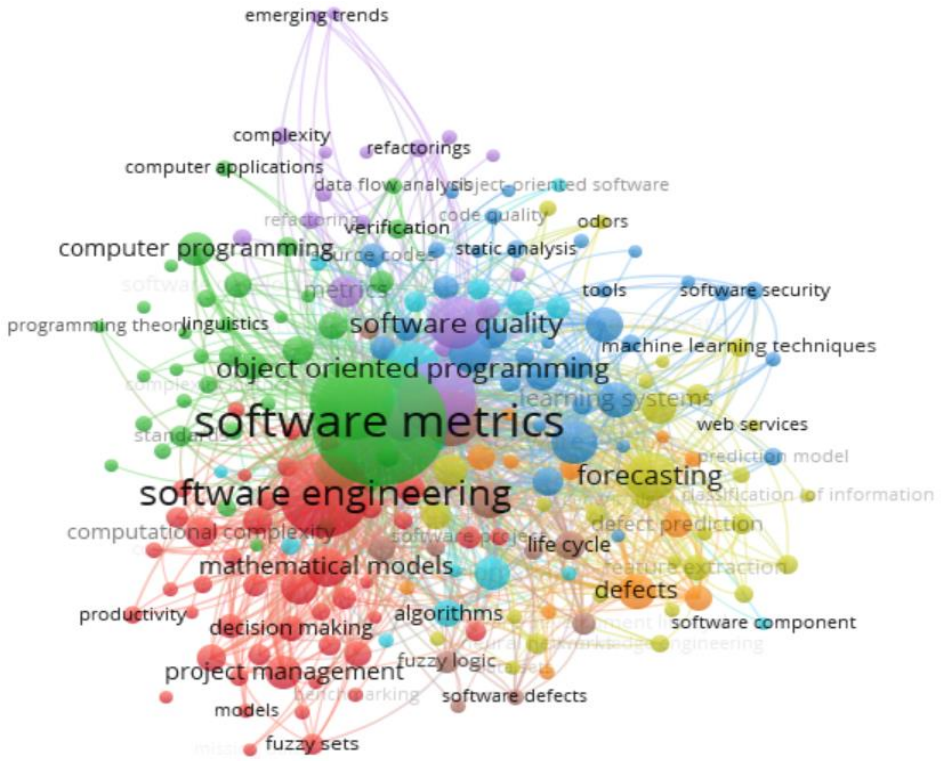

Figure 5: Keyword clusters.

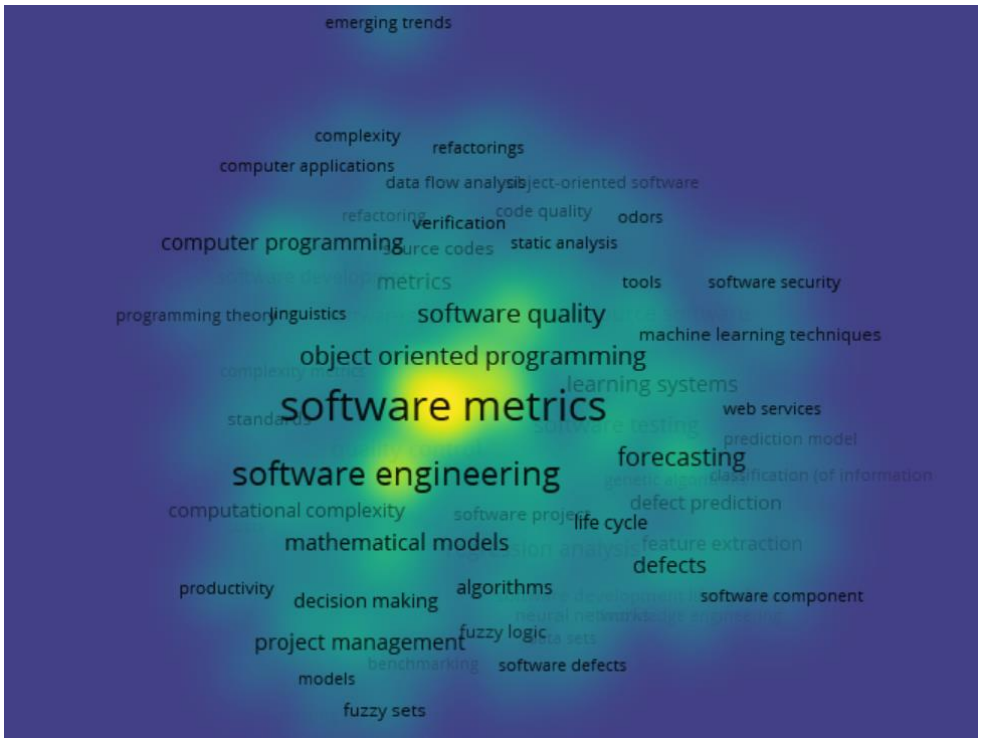

Figure 6: Density of keywords.

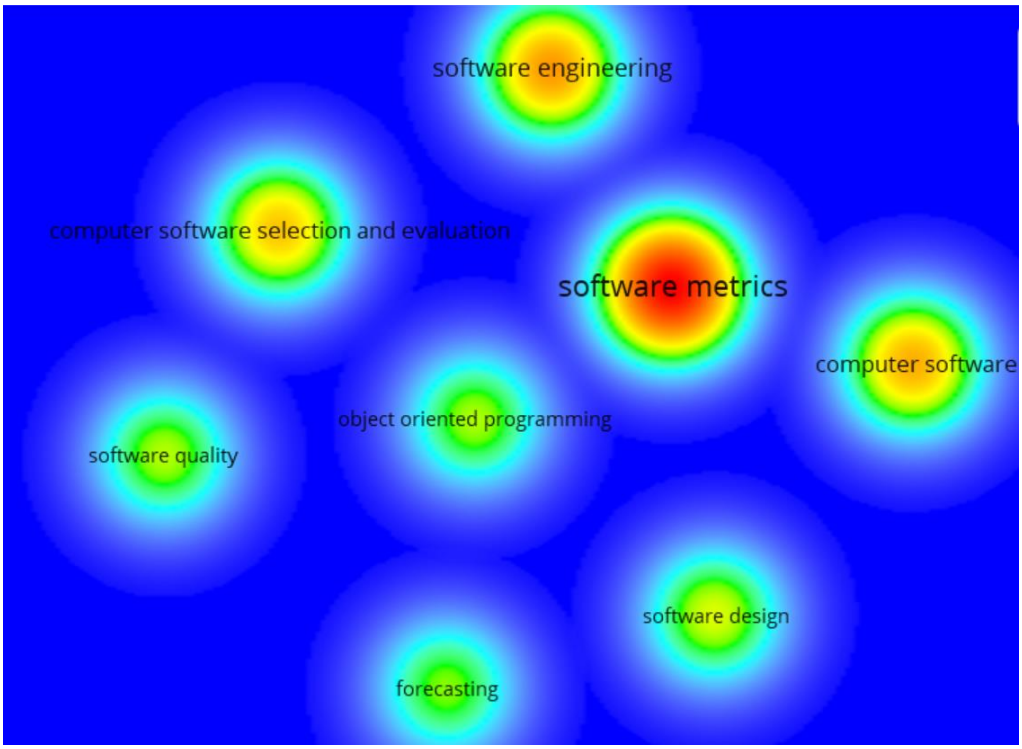

Figure 7: Top keywords.

Tables 2 and 3 show the number of publications by country in the Scopus and Web of Science databases, respectively. The United States ranks first in both databases with up to 249 publications. India ranks second with 108 publications, while Brazil ranks eighth with 31 publications in both databases.

Table 2: Index of publications by country (Scopus).

Country	Number of publications
United States	144
India	71
China	33
United Kingdom	30
Canada	29
Germany	27
Italy	24
Brazil	20
New Zealand	19
Japan	15
Netherlands	14
Australia	13
Spain	13
Slovenia	12
Sweden	12
Finland	11

Table 3: Index of publications by country (Web of Science).

Country	Number of publications
United States	105
India	37
England	25
Canada	23
Germany	20
China	19
Italy	12
Brazil	11
Netherlands	10

Tables 4 and 5 list the number of publications by area of knowledge. Computer science ranks first with 548 publications in the Scopus database and 321 publications in the Web of Science database, followed by the Engineering area with 178 publications in the

Scopus database and 110 publications in the Web of Science database. Publications in Computer Science and Engineering represent more than 80% of the total.

Table 4: Index of publications by area of knowledge (Scopus).

Knowledge area	Number of publications	Cumulative frequency
Computer science	**548**	**60,49**
Engineering	178	80,13
Mathematics	66	87,42
Decision Science	26	90,29
Business, Management and Accounting	23	92,83
Physics and Astronomy	16	94,59
Social Science	14	96,14
Materials Science	11	97,35
Energy	8	98,23
Chemical Engineering	4	98,68
Medicine	4	99,12
Environmental Science	3	99,45
Neuroscience	2	99,67
Multidisciplinary	2	99,89
Biochemistry, Genetics and Molecular Biology	1	100,00
Total	**906**	

Table 5: Index of publications by area of knowledge (Web of Science).

Knowledge area	Number of publications	Cumulative frequency
Computer Science	**321**	**62,57**
Engineering	110	84,02
Telecommunications	17	87,33
Business economics	7	88,69
Automotion control systems	7	90,06
Operations research management science	6	91,23
Mathematics	6	92,40
Science Technology Other Topics	6	93,57
Educational research	5	94,54
Imaging Science Photographic Technology	5	95,52
Optics	4	96,30
Information Science	3	96,88
Physics	3	97,47
Energy	2	97,86
Remote sensing	2	98,25
Robotics	2	98,64
Astronomy	1	98,83
Chemistry	1	99,03
Geology	1	99,22
Instrumentation	1	99,42

Materials science	1	99,61
Mathematical Methods in Social Sciences	1	99,81
Psychology	1	100,00
Total	**513**	

Tables 6 and 7 indicate the number of publications by journal, according to each database. The Journal of Systems and Software ranks first with 28 publications in the Scopus database and 25 publications in the Web of Science database. Journals with less than five publications were not mentioned.

Table 6: Indexes of publications by journals (Scopus).

Journals	Number of publications
Journal of systems and software	28
Lecture notes in computer science including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics	24
Proceedings international conference in software engineering	19
ACM international conference proceeding series	16
IEEE transactions in software engineering	14
Ceur workshop proceedings	13
Information and software technology	10
International software metrics symposium proceedings	10
ACM sigplan notices	9
Advances in intelligent systems and computing	7
Proceedings international computer software and applications conference	6
Proceedings of the acm symposium on applied computing	6
Proceedings of the European conference on software maintenance and reengineering csmr	6
Proceedings of the international conference on software engineering and knowledge engineering seke	6
Canadian conference on electrical and computer engineering	5
Proceedings ieee computer society s international computer software and applications conference	5
Proceedings international software metrics symposium	5
Proceedings international symposium on software reliability engineering issre	5
Software engineering journal	5
Software quality journal	5

Table 7: Indexes of publications by journals (Web of Science).

Journals	Number of publications
Journal of systems and software	25
Lecture notes in computer science	16
IEEE transactions on software engineering	13
Sigplan notices	12
Information and software technology	10
IEEE software	7
European conference on software maintenance and reengineering	5

Table 8 show the number of publications and the H-index by author. The author Khoshgoftaar, T.M. stands out with 17 publications in the Scopus database and five publications on the Web of Science database. In addition, he is the author with the highest H-index in both databases.

Table 8: Index of publications by authors, including the H-index.

Authors	Number of publications		H-index	
	Scopus	Web of Science	Scopus	Web of Science
Khoshgoftaar, T.M.	17	5	57	41
Wang, H.	11	----	13	----
Concas, G;	7	----	16	----
Heričko, M.	7	----	19	----
Schneidewind, N.F.	7	----	20	----
Tempero, E.	7	----	18	----
Napolitano, A.	6	----	24	----
Succi, G.	6	----	35	----
Mathiassen, L.	6	----	34	----
Allen, E.B.	5	----	24	----
Chan, V.K.Y.	5	----	4	----
Counsell, S.	5	----	26	----
Ejiogu, L.O.	5	5	2	2
Gray, A.R.	5	----	39	----
Sultana, K.Z.	5	5	7	1
Prather, R.E.	5	----	7	----
Wong, W.E.	5	----	32	----
Silitti, A.	5	----	28	----
Harrison,W.	4	5	10	3
Ince, D.	2	5	13	9
Pfleeger, S.L.	3	5	26	19

The VOSviewer software was used to analyse the author network. Figure 8 shows that there is no well-defined nucleus of authors on the subject, and that there are only level 3

links, with different authors. Of the 1,193 identified authors, 208 reached the minimum limit of two occurrences in publications.

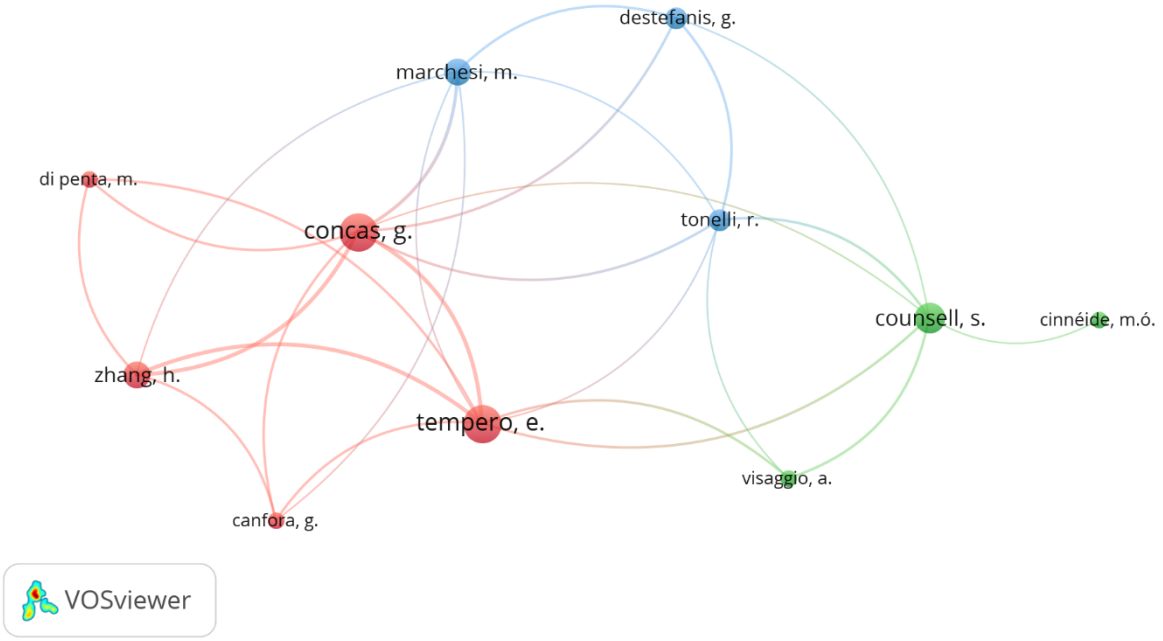

Figure 8: Author network (Scopus).

Tables 9 and 10 show the number of publications by affiliation, according to the databases. Florida Atlantic University stands out in the Scopus database with 16 publications, and the United States Department of Defence stands out in the Web of Science database with 7 publications. Institutions with less than five publications were not mentioned in the tables.

Table 9: Index of affiliation publications (Scopus)

Institution	Number of publications	Country
Florida Atlantic University	16	United States
Western Kentucky University	12	United States
Univerza v Mariboru	12	Slovenia
Università degli Studi di Cagliari	10	Italy
Brunel University London	8	United Kingdom
Univerza v Mariboru Fakultete za Elektrotehniko	8	Slovenia
IEEE	7	United States
Università degli Studi del Sannio	7	Italy
University of Auckland	7	New Zealand
University of Otago	7	New Zealand
University of Alabama in Huntsville	7	United States
International Business Machines	7	United States
Delft University of Technology	6	Netherlands

The University of Texas at Dallas	6	United States
Federal University of Minas Gerais	6	Brazil
Naval Postgraduate School	6	United States
University of Novi Sad	6	Serbia
National Research Council Canada	5	Canada
Aalborg University	5	Denmark
Vellore Institute of Technology	5	India
Technical University of Berlin	5	Germany
Beijing Normal University	5	China
Tsinghua University	5	China
Virginia Polytechnic Institute and State University	5	United States
University of New South Wales UNSW Australia	5	Australia
Macao Polytechnic Institute	5	Macao

Table 10: Index of affiliation publications (Web of Science).

Institution	Number of publications	Country
United States Department of Defense	7	United States
State University System of Florida	6	United States
University of Alabama Huntsville	6	United States
University of Alabama System	6	United States
City University London	5	United Kingdom
Open University UK	5	United Kingdom
Portland State University	5	United States
Technical University of Berlin	5	Germany
University of Cagliari	5	Italy
University of Maribor	5	Slovenia

5. SOFTWARE METRICS STUDIES

According to the scope of the research, 23 articles (Table 11) were selected for full reading to provide researchers and software developers with an initial set of information on software metrics.

Dey and Mockus (2020) investigated the relationship between the failures and the use of software to assess the quality of the software version. They measured the number of new users, the intensity of use, the frequency of use, exceptions and data, and the duration of the launch of mobile applications for Android and iOS for the Telecommunications area. They used the Bayesian Network and Random Forest models. The results showed that the number of new users is the main factor in determining the number of exceptions, and no direct links were found between the intensity and frequency of using software with software failures. The limitations are related to the type of software. Commercial software has closed source code, so it was not possible to choose which variables were measured.

Mhawish and Gupta (2020) developed an approach that uses model metrics to distinguish between standards and software design to reduce complexity and facilitate maintenance and evolution. The results show that the approach was effective.

Table 11: Selected articles.

Authors	Year	Paper title	Database
Dey, T.; Mockus, A.	2020	Deriving a usage-independent software quality metric.	Scopus
Mhawish, M.Y.; Gupta, M.	2020	Software Metrics and tree-based machine learning algorithms for distinguishing and detecting similar structure design patterns	Web of Science
Karanatsiou, D., et al.	2019	A bibliometric assessment of software engineering scholars and institutions	Scopus
Malhotra, R.; Sharma, A.	2019	Estimating the threshold of software metrics for web applications	Scopus Web of Science
Alqmase, M.; Alshayeb, M.; Ghouti, L.	2019	Threshold Extraction Framework for Software Metrics	Web of Science
Ludwig, J.; Xu, S.; Webber, F.	2018	Static software metrics for reliability and maintainability	Scopus Web of Science
Liu, Y., et al.	2018	Connecting software metrics across versions to predict defects	Scopus
Dósea, M.; Sant'anna, C.; Da Silva, B.C.	2018	How do design decisions affect the distribution of software metrics?	Scopus
Sheetal, A.P.; Ravindranath, K.	2018	Software metric evaluation on cloud based applications	Scopus
Boucher, A.; Badri, M.	2018	Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison	Scopus Web of Science
Mauša, G.; Grbac, T.G.	2017	The stability of threshold values for software metrics in software defect prediction	Scopus
Medeiros, N., Ivaki, N., Costa, P., Vieira, M.	2017	Software metrics as indicators of security vulnerabilities.	Scopus Web of Science
Aversano, L., et al.	2017	Investigating Differences and Commonalities of Software Metric Tools	Scopus
Amara, D.; Rabai, L.B.A.	2017	Towards a new framework of software reliability measurement based on software metrics	Scopus
Arvanitou, E.M., et al.	2016	Software metrics fluctuation: a property for assisting the metric selection process.	Scopus Web of Science
Rizvi, S.W.A.; Singh, V.K.; Khan, R.A.	2016	The state of the art in software reliability prediction: software metrics and fuzzy logic perspective	Scopus
Petkov, A.	2016	A software metric for the evaluation of testing efficiency	Scopus
Lakshmi, P.; Maheswari, T.L.	2016	An effective rank approach to software defect prediction using software metrics.	Scopus Web of Science
Bhardwaj, M.; Rana, A.	2016	Key Software Metrics and its Impact on each other for Software Development Projects	Scopus
Padmini, K.V.J.; Bandara, H.M.N.D.; Perera, I.	2015	Use of software metrics in agile software development process	Web of Science
Bozzelli, P.; Gu, Q.; Lago, P.	2013	A systematic literature review on green software metrics.	Scopus
Radjenović, D., et al.	2013	Software fault prediction metrics: A systematic literature review	Scopus
Lincke, R.; Lundberg, J.; Löwe, W.	2008	Comparing software metrics tools	Scopus

Dey and Mockus (2020) investigated the relationship between the failures and the use of software to assess the quality of the software version. They measured the number of new users, the intensity of use, the frequency of use, exceptions and data, and the duration of the launch of mobile applications for Android and iOS for the Telecommunications area. They used the Bayesian Network and Random Forest models. The results showed that the number of new users is the main factor in determining the number of exceptions, and no direct links were found between the intensity and frequency of using software with software failures. The limitations are related to the type of software. Commercial software has closed source code, so it was not possible to choose which variables were measured.

Mhawish and Gupta (2020) developed an approach that uses model metrics to distinguish between standards and software design to reduce complexity and facilitate maintenance and evolution. The results show that the approach was effective.

Malhotra and Sharma (2019) studied defect prediction using regression software metrics. They selected a set of object-oriented metrics as independent variables. The data were collected in the error reports of each project java file, in two consecutive releases of the software, and were converted into binary values to be used in the regression. As such, the data considered as defective received value "1", and the non-defective data "0". The regression was calculated considering two hypotheses: H_0: the calculated values cannot distinguish between defective and non-defective classes (null hypothesis); H_1 the calculated values can distinguish between defective and non-defective classes (alternative hypothesis). If the calculated limit value had low significance (<0.05), the null hypothesis is rejected. After performing the calculations, six of the sixteen previously selected metrics were classified as of vital importance for the prediction of defects (WMC, RFC, NPM, LCOM3, MOA, and LOC).

Alqmase et al. (2019) proposed a framework based on the expectation maximization (EM) algorithm in which clusters are generated using a simplified set of metrics (LOC, LCOM, and CBO). The limits of the clusters are calculated by Source Code Analysis and Manipulation using statistical measures, such as mean and standard deviation. The results highlight the structure's ability to group software quality data sets according to different quality levels.

Karanatsiou et al. (2019) conducted a bibliometric analysis of software engineering scholars and institutions. They used the mapping study technique on top-quality software engineering venues and developed a dataset of 14,456 primary studies.

Ludwig et al. (2018) created an open-source plug-in to identify a set of static software code metrics tied to the reliability and maintainability quality characteristics of the software product and technical debt sources. The execution of the plug-in developed by the authors performs a static analysis of the software code and generates a report in the HTML format containing the metrics found and the groupings in relation to the architecture. The authors conclude that the main avoidable technical debts are architectural decisions, overly complex code, and missing code documentation.

Liu et al. (2018) proposed the use of historical versions of sequence metrics (HVSM) in continuous versions of software to predict defects using a recurrent neural network. The

evaluation was carried out in nine open-source projects, and the results showed that the model proposed by the authors has significantly better classification efficiency than the models commonly used in seven of the projects considered in the study.

Dósea et al. (2018) conducted a study in the source code of fifteen systems from three different domains to evaluate the distributions of four metrics (CC; NMP; LOC; EC) in relation to software design functions. The results show that the distribution of metrics is sensitive to the following design decisions: (i) design function of class libraries (ii) libraries used, (iii) coding style, (iv) exception handling and (v) mechanism registration and code debugging.

Sheetal and Ravindranath (2018) proposed a new set of metrics for use in cloud-based software applications since they understood there is a significant demand for it. The proposed model considers metrics related to architecture model attributes and software design and can be used in applications that have 40 to 200 modules. Data was collected in 15 random applications. The authors conclude that the new set of metrics provides a new generic trend of analysing applications that migrate to the cloud, thereby meeting the demands of modern research and customer satisfaction.

Boucher and Badri (2018) investigated the performance of seven source code metrics in predicting failures. Data were collected from 12 public databases. The authors used three threshold definition methods to predict the susceptibility to failure: ROC Curves, VARL and Alves Rankings. The useful software XLSTAT was used to perform the univariate logistic regression, in which the metrics are the independent variables. The results show that (i) the SLOC, CBO, RFC, WMC and LCOM metrics are relevant for the prediction of fault propensity in most of the datasets investigated, (ii) ROC curves are the best performing method among the three investigated.

Mauša and Grbac (2017) constructed a univariate logistic regression model for fifty metrics, employing the Matlab software to predict failures and improve software quality. In addition, the authors used the BuCo Analyser tool to collect data from 14 datasets belonging to two open-source projects. The metrics that had a value of $p > 0.05$ were not considered relevant for the prediction of defects. A statistical significance test was used to cross-validate the geometric mean precision of the defect prediction. The results revealed that all metrics that passed the two criteria of significance had stable threshold values within the same version of a project. In short of the fifty metrics considered by the authors, 26 are significant in predicting defects at each launch in both projects; eight of them are rarely significant in both projects; and six of them are significant in one project but not the other.

Medeiros et al. (2017) performed a software metrics analysis at different architecture levels of five software projects to see if metrics can classify vulnerable and non-vulnerable code units. The programming languages of the projects were C and C ++; in addition, a large set of software metrics of different types were considered, including complexity, volume, coupling and cohesion metrics, totalling 28 function-level metrics; 51 file-level metrics, and 54 project-level metrics. The results indicate that i) there is a strong correlation between several metrics at the project level and the number of reported vulnerabilities; ii) it is possible to use a metric group to distinguish between vulnerable and non-vulnerable code units with a high level of accuracy; iii) the best subset of predictive metrics varies from system to system;

and iv) to identify the quality of the software in terms of security, the function, file and project-level metrics are complementary to each other.

Aversano et al. (2017) investigated the behaviour of software metrics tools on the market. The authors considered the following software quality metrics (i) six object-oriented metrics; (ii) 11 metrics relative to software size; (iii) two in relation to complexity, (iv) in addition to five other metrics. The comparison was made among nine tools, using data referring to three open-source software systems of different sizes (small, medium, large) and written in Java. The objective was to verify if the analysed tools interpret the same set of metrics in the same way. The results show that each tool calculates a set of different metrics, and only the LOC metric is calculated by all tools. Another important point of the research is that the values calculated for each metric show differences from one tool to another, which may occur due to the size of the software.

Amara and Rabai (2017) conducted a review of the methods used to measure software reliability and proposed a new approach for this measurement based on software metrics that can be used at each stage of the software development lifecycle (SDLC), therefore producing reliable software instead of evaluating the reliability of the finished product. In the proposed approach, the use of reliability models and techniques starts in the Requirements Elicitation phase, aiming to improve reliability, developing the necessary specifications, and avoiding unnecessary tests. With this, fixes and improvements can be implemented at lower costs and in less time. In addition, the use of software metrics, models and reliability techniques in the testing phase can increase reliability while minimizing the cost and effort required for patches and improvements.

Arvanitou et al. (2016) define buoyancy as the degree to which a metric is capable of capturing changes in the structure of the underlying version of a software system. They investigated the fluctuation of 19 object-oriented metrics, considering cohesion, complexity, inheritance, coupling, and size. Twenty open-source projects were selected to verify metric fluctuation. The results show that (i) the number of methods increased in approximately 75% of transitions from one version to another while remaining stable in about 13%; (ii) source code metrics are more sensitive than project-level metrics.

Rizvi et al. (2016) conducted a study to verify the state of the art from the perspective of software metrics and fuzzy logic. The authors highlight some important points from the literature review: (i) in the absence of failure data during the early stages of product development, appropriate software metrics can help predict reliability; (ii) the existence of predictive models based on the probabilistic approach, and the fact that it is not possible to define some measures precisely, justify the use of a fuzzy logic approach to predict software defects; (iii) proper selection of software metrics is of paramount importance; (iv) the ability to predict software reliability in advance provides vital information for decision making about design and resource allocation. The authors conclude the paper by stating that the critical analysis of models to predict software failures and reliability revealed that software metrics, together with fuzzy logic techniques, produce better results in predicting software reliability and residual defects.

Petkov (2016) presents a metric for assessing the efficiency of the software testing process based on the number of bugs discovered during testing and by customers. A case study was conducted of four projects within an organization. The evaluation of the efficiency

of the test metrics was carried out according to the activities defined in the CMMI measurement process to consider a metric applicable: (i) defining the information needs that the metric satisfies; (ii) defining the data collection method and the necessary measures; (iii) performing the measurements in the projects under investigation; (iv) analysing the correlation between metric values and information needs. A good testing process uncovers errors as quickly as possible, which is why the testing process runs in parallel with the development process and starts right away in defining the software requirements. Finding bugs early reduces the lifecycle, making software cheaper. The purpose of the tests is to discover and remove errors before they are sent to users. The author concludes that the "test efficiency metric" provides a satisfactory measure of the effectiveness of the test process.

Lakshmi and Maheswari (2016) performed defect prediction using object-oriented metrics and software version history. The Out-Of-Bag (OOB) approach was used to rank the most effective attributes from the information set for predictions. Once the prediction method is completed, the square measurement of the modules is stratified according to its severity, making it possible to calculate the cost of correcting the problem. The authors conclude that predicting an explicit range of defects for each code module is a difficult task, and if there is a shortage of correct historical information, it is not possible to perform the prediction.

Bhardwaj and Rana (2016) explain the relationship between key metrics (size, effort, duration, and productivity) and how they statistically affect each other, in addition to predicting the total number of software defects. The relationship between these metrics can be derived using historical data from similar projects of the organization, but only if the organization process has a high maturity level, otherwise the data repository is not enough to reveal any statistical relationship. The results show that (i) metrics related to software size significantly affect the other three (effort, duration, and productivity); (ii) productivity does not depend significantly on software size but represents the nonlinear relationship with software size and maximum team size; (iii) the total duration of the project depends only on the size of the software and does not depend on the maximum size of the team. The authors recommend that all metrics be recalibrated during the project development lifecycle.

Padmini et al. (2015) conducted interviews in 24 companies to identify software metrics for agile software development, since the metrics used in traditional development do not apply. They identified metrics that can bring benefits to agile software development, with a focus on product quality, team productivity and predictability.

Bozzelli et al. (2013) proposed a division of software metrics in three groups: measured resources, kind of results (type), environment or application domain (context), and usage.

Radjenovic´ et al. (2013) conducted a study to identify which software metrics are most used to predict software failures. The results showed that object-oriented metrics (49%) were used almost twice as much as sources code metrics (27%) or process metrics (24%).

Lincke et al. (2008) presented a comparison of software metrics tools: CBO (Coupling Between Object classes), DIT (Depth of Inheritance Tree), LCOM-CK (Lack of Cohesion of Methods), LCOM-HS (Lack of Cohesion of Methods), LOC (Lines of Code), NOC (Number Of Children), NOM (Number Of Methods), RFC (Response For a Class), WMC (Weighted Methods per Class). They concluded that, for rather simple metrics, like the

Number of Children (NOC), most tools computed the same or remarkably similar results. For other metrics, e.g., the Coupling Between object Classes (CBO) or Lack of Cohesion of Methods (LCOM), the results showed a much bigger variation.

6. NEW PROPOSAL

The literature review was carried out from the bibliometric study to identify which types of software metrics have been used by the developers. The results showed that most metrics are used to measure aspects related to the source code, none were related to acceptance.

It is also noticed that many researchers are proposing new metrics every year to obtain some gain such as quality, for example. This fact indicates that, although there are already many types of software metrics in the literature, there are still areas in different phases of the software development process that lack them.

In this context, we propose a support metric for software acceptance (MESSA) to facilitate the delivery phase of the software product and, thus, provide security to the customer and cost savings for the developing company, since the difficulties encountered at this stage can make the process more expensive for the company.

The MESSA methodology includes four phases (Figure 9):

(i) Creation of a use case diagram, according to the requirements and business rules specified in the contract. A CASE tool will be used to create the diagram and export the XMI file;

(ii) An application, developed for this purpose, will read the XMI file, which will generate a report with the use case information.

(iii) Customers define the importance of each use case, and the preference of the development order. A multicriteria method will be used to classify the CDUs.

(iv) Update of the software acceptance document at each iteration, through the customer's evaluation. Use of the software acceptance document when delivering the software.

The proposed methodology must be used throughout the software product development process. When contracting the service, the customer establishes the requirements and business rules of the software.

Initially, an acceptance report is presented to the customer, so that he can establish the importance and priority for each use case. The next step is to use the multicriteria method to identify the development order.

Thereafter, at each iteration, this acceptance report is presented to the customer to establish the percentage of acceptance of the use cases already developed. As well as

informing priorities. Provision of update of the acceptance report. As well as informing priorities, providing update of the acceptance report.

The software product is considered ready for delivery, when the report shows that all use cases have been 100% met.

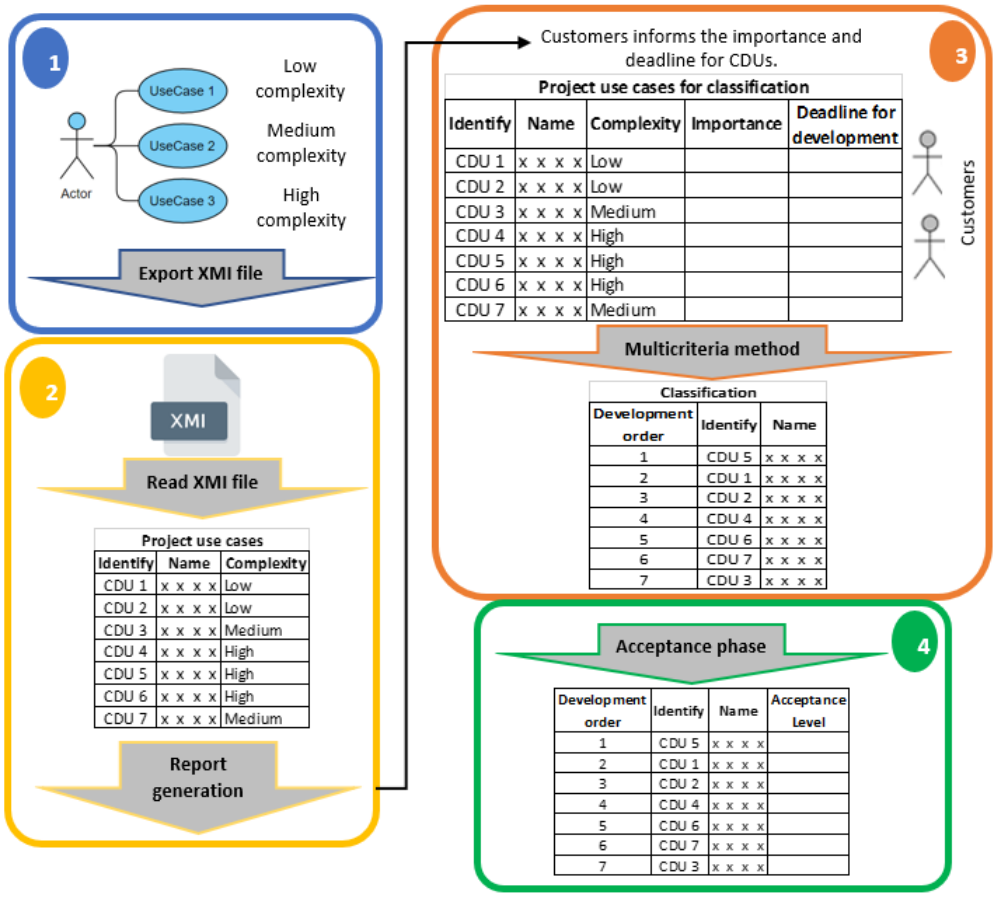

Figure 9: MESSA methodology.

7. CONCLUSIONS

This paper aimed to propose a new software metric based on a bibliometric study and a literature review on software metrics.

The literature review reveals that there is no set of defined metrics. Researchers are still proposing new software metrics to meet the needs of specific types of software. This context shows the importance of using metrics to improve the quality of the software, as well as the productivity of the team. It is important to highlight that, according to the parameters defined for bibliometrics, no software acceptance metrics were identified.

Some types of software metrics stand out in the articles selected for the literature review. All of them are related to the software code: CBO (Coupling Between Object classes); DIT (Depth of Inheritance Tree); LCOM-CK (Lack of Cohesion of Methods); LCOM-HS (Lack of Cohesion of Methods); LOC (Lines of Code); NOC (Number of Children); NOM (Number of Methods); RFC (Response For a Class); WMC (Weighted Methods per Class); NPM (Number of Public Methods); MOA (Measure of Aggregation). This is an important aspect of software development because it indicates that code is the most controlled part by companies.

Although there are investments in the IT sector in Brazil and a growing number of companies focused on software development and marketing, the number of publications on the subject is still timid.

As limitation, these results cannot be generalized since the bibliometric study considered only two databases. In this sense, other databases could be studied in future works to observe the evolution of scientific production on software metrics.

In addition, a case study will be carried as future work to apply the MESSA proposal. The methodology will be tested by a team that uses the development of hybrid software products and then it will be used by traditional and agile teams.

After using the methodology by these teams, the results will be compared, verifying the difficulties encountered and the suggestions for the improvement of the methodology.

Acknowledgments: This study was supported, without funding, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES).

8. REFERENCES

ABES. (2020). **Brazilian Association of Software Companies**. Available at: <http://www.abessoftware.com.br>. June 2020.

ABNT: **Brazilian Association of Technical Standards**. NBR ISO/IEC 9126-1: Software Engineering - Product Quality". Part 1: Quality Model, 2003.

ALQMASE, M.; ALSHAYEB, M.; GHOUTI, L. Threshold Extraction Framework for Software Metrics. **Journal of Computer Science and Technology**, v.34, n.5, p.1063-1078, 2019.

AMARA, D.; RABAI, L.B.A. Towards a new framework of software reliability measurement based on software metrics. **Procedia Computer Science**, v.109, p.725-730, 2017.

ARVANITOU, E.M. et al. Software metrics fluctuation: a property for assisting the metric selection process. **Information and Software Technology**, v.72, p.110-124, 2016.

AVERSANO, L. et al. Investigating Differences and Commonalities of Software Metric Tools, 2017, In: **ICSOFT**. p.249-256.

BERANIČ, T.; HERIČKO, M. Comparison of systematically derived software metrics thresholds for object-oriented programming languages. **Computer Science and Information Systems**, v.17, n.1, p.181-203, 2019.

BHARDWAJ, M.; RANA, A. Key Software Metrics and its Impact on each other for Software Development Projects. **ACM SIGSOFT** *Software Engineering Notes*, v.41, n.1, p.1-4, 2016.

BOUCHER, A.; BADRI, M. Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison. **Information and Software Technology**, v.96, p.38-67, 2018.

BOZZELLI, P.; GU, Q.; LAGO, P. **A systematic literature review on green software metrics**, 2013, VU University, Amsterdam.

COSTA. H.G. Model for webibliomining: proposal and application. **Revista da FAE**, v.13, n.1, p.115-126, 2010.

DA SILVA, F.E. et al. Uso de software como suporte às atividades de gestão da qualidade. **Exatas & Engenharias**, v.9, n.26, p.45-54, 2019.

DE PÁDUA PAULA FILHO, W. Engenharia de software [**Software Engineering**]. LTC, 2003.

DEY, T.; MOCKUS, A. Deriving a usage-independent software quality metric. **Empirical Software Engineering**, v.25, n.2, p.1596-1641, 2020.

DÓSEA, M.; SANT'ANNA, C.; DA SILVA, B.C. How do design decisions affect the distribution of software metrics? In: *Proceedings of the 26th Conference on Program Comprehension*. **ACM**, 2018, p.74-85.

FENTON, N.; BIEMAN, J. **Software metrics**: A rigorous and practical approach. CRC Press, 2014.

FREITAS, N.N.; MONTEIRO, S.B.S. Simulação computacional como ferramenta de suporte a decisão. **Exatas & Engenharias**, v.*7*, n.17, p.98-113, 2017.

FREITAS, R.X.; ALBUQUERQUE, A.B.; SANTOS, R. A utilização de guias, normas e modelos de maturidade como apoio à aquisição e contratação de tecnologia da informação em instituições públicas brasileiras [The use of guides, standards and maturity models to support the acquisition and hiring of information technology in Brazilian public institutions]. In: *XIII Workshp anual do MPS (**WAMPS**)*, 2017.

HASSAN, R. et al. Usability Requirements Extraction Method from Software Document. **International Journal of Software Engineering and Knowledge Engineering**, v.30, n.2, p.171-189, 2020.

ISHIKIRIYAMA, C.S.; MIRO, D.; GOMES, C.F.S. Text Mining Business Intelligence: A small sample of what words can say. **Procedia Computer Science**, v.55, p.261-267, 2015.

KARANATSIOU, D. et al. A bibliometric assessment of software engineering scholars and institutions (2010–2017). **Journal of Systems and Software**, v.147, p.246-261, 2019.

LAKSHMI, P.; MAHESWARI, T.L. An effective rank approach to software defect prediction using software metrics. In: *10th International Conference on Intelligent Systems and Control (ISCO)*. 2016, **IEEE**, p.1-5.

LINCKE, R.; LUNDBERG, J.; LÖWE, W. Comparing software metrics tools. In: *Proceedings of the 2008* ***international symposium on Software testing and analysis***. 2008. p.131-142.

LIU, Y. et al. Connecting software metrics across versions to predict defects, 2018, In: *IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (**SANER***). IEEE, p.232-243.

LUDWIG, J.; XU, S.; WEBBER, F. Static software metrics for reliability and maintainability. In: ***IEEE/ACM*** *International Conference on Technical Debt (TechDebt*). IEEE, 2018. p.53-54.

MALHOTRA, R.; SHARMA, A. Estimating the threshold of software metrics for web applications. **International Journal of System Assurance Engineering and Management**, p.1-16, 2019.

MAUŠA, G., GRBAC, T.G. The stability of threshold values for software metrics in software defect prediction, 2017. In: ***International Conference on Model and Data Engineering. Springer***, Cham. p.81-95.

MEDEIROS, N. et al. Software metrics as indicators of security vulnerabilities. In: *IEEE 28th International Symposium on Software Reliability Engineering* (**ISSRE**), 2017, IEEE p.216-227.

MELLUZZI NETO, G. et al. Resultados da implantação de CMMI e MPS-BR em empresas de desenvolvimento e manutenção de software: a visão da alta gestão [Results of the implementation of CMMI and MPS-BR in software development and maintenance companies: the view of top management]. **Revista Brasileira de Computação Aplicada**, v.10, n.1, p.2-10, 2018.

MHAWISH, M.Y.; GUPTA, M**.** Software Metrics and tree-based machine learning algorithms for distinguishing and detecting similar structure design patterns. **SN Applied Sciences**, v.2, n.1, p.11, 2020.

MISIRLIS, N.; VLACHOPOULOU**,** M. Social media metrics and analytics in marketing – S3M: A mapping literature review**. International Journal of Information Management**, v.38, n.1, p.270-276, 2018.

PADMINI, K.V.J.; BANDARA, H.M.N.D.; PERERA, I. Use of software metrics in agile software development process. In: *Moratuwa Engineering Research Conference (**MERCon***). IEEE, 2015. p.312-317.

PETKOV, A. A software metric for the evaluation of testing efficiency. In: ***AIP** Conference Proceedings.* AIP Publishing, 2016, p.060001.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software**. 8ª Edição. McGraw Hill Brasil, 2016.

RADJENOVIĆ, D. et al. Software fault prediction metrics: A systematic literature review. **Information and software technology**, v.55, n.8, p.1397-1418, 2013.

RIZVI, S.W.A.; SINGH, V.K.; KHAN, R.A. **The state of the art in software reliability prediction: software metrics and fuzzy logic perspective**. In: *Information Systems Design and Intelligent Applications*. Springer, New Delhi, 2016, p.629-637.

SHEETAL, A.P.; RAVINDRANATH, K. **Software metric evaluation on cloud based applications**. In: *International Journal of Engineering & Technology*, 2018, v.7, p.13-18.

SILVA, G.B.; COSTA, H.G.; BARROS, M.D. Entrepreneurship in engineering education: A literature review. International Journal of Engineering Education, v. 31, n.6A, p.1701-1710, 2015.

SOFTEX. (**Brazilian Software Excellence Promotion Association**). MPS.BR– Brazilian Software Process Improvement: General MPS Software Guide, 2016.

SOMMERVILLE, I. **Engenharia de software** [Software engineering]. 10ªEd. Pearson Prentice Hall, 2019.

TEMPERO, E.; RALPH, P. A framework for defining coupling metrics. **Science of Computer Programming**, v.166, n.15, p.214-230, 2018.

TRIPATHI, M.K. et al. Prediction of Quality of Service Parameters Using Aggregate Software Metrics and Machine Learning Techniques. In: *2018 15th IEEE India Council International Conference (**INDICON**). IEEE*, 2018. p.1-6.

TUMMALAPALLI, S.; KUMAR, L.; MURTHY, N.L.B. Prediction of Web Service Anti-patterns: Using Aggregate Software Metrics and Machine Learning Techniques. In: ***Proceedings of the 13th Innovations in Software Engineering Conference** on Formerly known as India Software Engineering Conference*, 2020. p.1-11.

VOGEL, R.; GÜTTEL, W.H. The Dynamic Capability View in Strategic Management: A Bibliometric Review**. International Journal of Management Reviews**, v.15, p.426-446, 2013.

WAZLAWICK, R. **Engenharia de software: conceitos e práticas** [Software engineering: concepts and practices]. Elsevier Editora Ltda., 2019.

ZHANG, F.; WU, S. Predicting future influence of papers, researchers, and venues in a dynamic academic network. **Journal of Informetrics**, v.14, n.2, p.101035-101059, 2020.

ZHU, J.; LIU, W. A tale of two databases: the use of Web of Science and Scopus in academic papers. **Scientometrics**, p.1-15, 2020.