An Analysis of Measurement and Metrics Tools: A Systematic Literature Review

Edna Dias Canedo
Computer Science Department
University of Brasília (UnB)
P.O. Box 4466, 70910-900, Brasília-DF, Brazil
ednacanedo@unb.br

Karine Souza Valença
Faculty UnB Gama (FGA)
University of Brasília (UnB)
Brasília-DF, Brazil
valenca.karine@gmail.com

Giovanni Almeida Santos
Faculty UnB Gama (FGA)
University of Brasília (UnB)
Brasília-DF, Brazil
giovannix@unb.br

Abstract

Measurement is an important field in Software Engineering, since it allows for organizations to obtain trustworthy estimates regarding deadlines, cost, and quality for the development of their software projects. Many tools are available for the calculation and storage of metrics and therefore, choosing the best tool can be a hard task. Faced with such a problem, this study carries out an analysis of the measurement tools presented in literature. The methodology chosen for the task was the systematic literature review. The results of the systematic review present the metric tools chosen in literature, their functionalities, and the main metrics used by these tools. The primary contribution of this article is a list with the metrics used by each of these tools, and their respective classification, according to their use in academia as well as in the software industry.

1. Introduction

Software metrics continue to be of interest for researchers and practitioners. Software metrics are an essential aid to the software measurement process and new software metrics continue to be defined by the research community. Software measurement is a task that is carried out during all the phases of the software development process. During this process, many intermediate or final software products are developed and measured by software product metrics. One of those products is the project management (scope, time, cost and quality), which is part of the final software system and is measured to assess its quality. This measurement is achieved by mapping a particular characteristic of a measured entity to a numerical value, taking in consideration that for some cases that value can be also categorical or ordinal [1].

Measurement, when done effectively, provides enough knowledge for software development organizations to make estimates in a trustworthy way and detect issues ahead of time. This allows them to have better control over risk management, and improves the overall quality of their products and projects [2]. The properties and quality of a measurement scale can be assessed by evaluation of the scale's reliability, construct validity and content validity [3].

The evaluation of a new metric typically consists of correlating the (change in) value of the metric with other quality indicators such as likelihood of change or its ability to predict the lack of planning. In other cases, the evaluation consists of an analysis of the values of a metric for a set of systems, either on one single snapshot or over a period of time. The organizations use different metrics to evaluate their products and processes. These metrics can be related to size, productivity, quality, and complexity, among others [4], [5]. As such, it is useful for the organization to have methods and tools to collect the metrics. It is also important that the registering, calculation, and presentation of such metrics be done in an automated way, since it allows for more agility in collecting and analyzing data, as well as minimizing possible annotation errors [4].

In the literature, there are several tools available for the calculation and storage of metrics. As a result, choosing the most adequate tool for the software development organization can be an arduous, complex task. To remedy this issue, this study seeks to make an analysis regarding measurement tools found in the literature. In this paper we evaluate and analysis of measurement and metrics tools present in literature, their functionalities, and the main metric used by these tools.

The remainder of the paper is organized as follows. Section 2 presents the theoretical basis required for the understanding of the study. Section 3 presents the methodology used for the development of the study. Section 4 presents the results obtained through the systematic literature review. Section 5 presents the discussion of the obtained results. Finally, Section 6 presents the conclusions and expectations of future studies.

URI: https://hdl.handle.net/10125/60133
ISBN: 978-0-9981331-2-6

2. Background

Measurement in the domain of Software Engineering corresponds to the successive process of defining, collecting, and analyzing data in the software development process, to understand and control the processes [6]. That is, through measurement, several useful items of information are studied and analysed and, through them, one can discover how the process is executed, what results are being generated in it, and also learn about managing the process, making the process better.

There are many definitions of what measurement is. One of the classic definitions is presented by Finkelstein & Leaning [7], who says that measurement is the ascribing of numbers to the properties of objects of real world events through an objective empirical operation, so as to describe them. In the study presented by Fenton and Pfleeger [4], measurement is classified as the process through which numbers and symbols are ascribed to attributes of real world entities, so as to describe them according to clearly defined rules. Based on these basic concepts, it can be said that measurement seeks to numerically characterize the attributes of objects and real world events.

The work presented by Galin [8] states that a measurement is crucial to the progress of all sciences. Scientific progress is made through observations and generalizations based on data and measurements. It is no different in Software Engineering, and measurements have a very important role to play in the success of organisations, as such measurement processes allow the software teams come to grips with their capacities. As a result, with measurements it is possible to carry out solid project planning that will not overshoot the planning done as regards the scope, quality, risk, and project length, as the measurement allows one to gain knowledge on the processes [2]. Apart from that, there are many characteristics in software projects that can be measured [9], strengthening the importance of measurement in this area.

Software metrics are a way of putting a value/measure on certain aspects of development allowing it to be compared to other projects. These values have to be assessed correctly otherwise they will not give accurate measurements and can lead to false estimations, etc. Metrics are used to maintain control over the software development process. It allows managers to manage resources more efficiently so the overall team would be more productive. Some examples of metrics include Size Projections like Source Byte Size, Source Lines of Code, Function pointers, GUI Features and other examples are Productivity Projections such as Productivity Metrics [10].

The metrics can be used to measure size, cost, effort, product quality of a project as well as putting a value on the quality of the process taken and personal productivity. There are certain factors that have to be taken into account when comparing different projects with each other using metrics. If one project has was written in a different language then the number of lines of code could be significantly different or perhaps the larger project has many more errors and bugs in it. Measurements such as Function Pointers give a better indication because they are the actual methods that are in the project rather than the number of lines [4].

A measurement scale corresponds to an ordered set of values or categories, which will map a database that stores the results of measurements and lessons learned [10]. A measurement scale can be classified in five main types [4], [11]: Nominal; Ordinal; Interval; Absolute, Ratio. **Nominal**, in this type of scale, there is no notion of ordering among the classes that define the attribute. Besides that, there is no magnitude. The values of measurement are basically categorical. **Ordinal**, in this type of scale, there is the notion of ordering, where measurement values signify position. Mathematical operations in this kind of scale are not used. **Interval**, in this type of scale, there is the notion of ordering, where measurement values keep equal distances, but not proportions. Mathematical operations such as additions and subtractions are accepted in this type, but multiplications and divisions are not allowed. **Ratio**, in this type of scale, there the notion of ordering and the interval between values. Besides that, proportion is also kept. There is the zero value, which corresponds to no attribute. All arithmetic operations are allowed in this type of scale. **Absolute**, in this type of scale, measurement is done only through counting the number of elements of the entity. No arithmetic operation is done in this type of scale.

3. Systematic Literature Review

Systematic literature review (SLR) is a secondary form of study that looks to identify and analyze research relevant to a certain research question. Among its characteristics, we can highlight [12]: It has a SLR process that is defined at the start of the research. This protocol defines the research questions to be approached, and which methods will be used during the research; It proposes the creation of a well-documented research strategy. This strategy needs to be good enough to get the highest amount of primary studies in literature relevant to the topic; It uses inclusion and exclusion criteria to evaluate the primary studies found.

The systematic review used in this study was composed of three main stages, as defined in [12]. The objective of the stages of the systematic review encompass: **Stage 1** - **Planning the review:** In this stage the research questions addressed in this study were chosen. Besides that, the SLR process to be used was developed. **Stage 2** - **Conducting the review:** In this stage, the SLR process defined in the previous stage was applied in the systematic literature review. **Stage 3** - **Reporting the review:** In this stage, the results obtained through the systematic review were revised and documented.

3.1. Research Questions

Seeking to reach the objective of this study, three research questions (RQ) were determined. These questions attempt to direct the searches and showcase the current overview of research done in the measurement tools field. The defined research questions were: **RQ.1 – Which measurement tools exist in the literature? RQ.2 – What functionalities do the measurement tools offer? RQ.3 – Which metrics were identified and used by measurement tools?**

3.2. Chosen SLR Main Activities

This systematic literature review strictly followed a SLR process. The activities for the systematic review are:

1. **Apply a search string in the research bases**: Apply the defined search string in the research bases. This activity will be better detailed in Section 3.3.

2. **Read the title, abstract, and keywords**: Read this information in the article, in order to verify if it fits the established inclusion and exclusion criteria.

3. **Apply inclusion and exclusion criteria**: Observe the inclusion and exclusion criteria and verify if the article fits them. It should otherwise be discarded. Section 3.4 presents the criteria defined for this review.

4. **Perform a thorough reading of the article**: Verify if the article answers the research question and if it's within the inclusion and exclusion criteria. If the article does not answer the research question, it must be discarded after being read.

5. **Extract the data**: Perform the collection of data that answers the research question and found in the selected articles.

6. **Fill the form for data collection and analysis**: Systematically organize all the data obtained during the systematic review, and create a detailed report of the data. Section 3.6 presents the result of this activity.

7. **Report the obtained results**: Answer the defined research questions based on the articles selected during the review procedure.

3.3. Search strategy

To perform a search in the research bases, the **search string** was created. The string was defined using the PICO **(Population, Intervention, Comparison, Outcomes)** approach, which helps in better framing the research questions. Each letter of the PICO approach corresponds to the following [12]: 1. **Population:** The population corresponds to a specific role within the lifecycle of the software, an area of application, or even a specific group of the industry; 2. **Intervention:** The intervention corresponds to methodologies, tools, technologies, or software procedures that broach a question; 3. **Comparison:** Comparison corresponds to methodologies, tools, technologies, or software procedures in which intervention is being compared. A comparison isn't always mandatory; 4. **Outcomes:** The outcomes must report important factors relevant to the professionals of the software field.

According to this definition, to answer the research questions, the defined PICO was: 1. **Population:** Software measurement field. 2. **Intervention:** Metrics collection tool. 3. **Comparison:** Does not apply. 4. **Outcomes:** Functionality. After executing the PICO approach, the following **search string** was obtained:

(TITLE-ABS-KEY ((”software measurement” OR ”software metrics”)) AND TITLE-ABS-KEY ((”metric collection tool” OR ”metric application” OR ”metric tool” OR ”metrics tool” OR ”metrics collection tool” OR ”software tool” OR ”metrics tools” OR ”measurement tools” OR ”measurement tool”)))

With the search string defined, automatized searches were performed in the SCOPUS digital library, since SCOPUS indexes different research bases and is one of the bases that indexes the main journals of Software Engineering, including: Digital Library IEEE Xplore; Digital library ACM; Science Direct. Scopus also indexes other relevant bases. The full list of indexed sources can be verified on the SCOPUS website[1]. Besides that, automated searches were also performed in the Science Direct base, since the Scopus indexing wasn't returning all the results for this specific base.

[1]https://www.scopus.com/sources?zone=&origin=sbrowse

3.3.1. Snowballing

Database searches are challenging for a variety of reasons, including selecting databases to use, different interfaces to databases, different ways of constructing search strings, different search limitations in databases, and identifying databases and synonyms of terms used [13]. This reasoning leads to two conclusions: 1. Choosing the first step in the search strategy often becomes the only step, that is, search databases; 2. Given the challenges with the databases, important studies can be lost.

Based on the snowballing instructions proposed by Wohlin and Prikladniki [13], in this study the steps used to perform this technique were: 1. Use the papers selected in automatic and manual searches as the initial set of selected studies; 2. Based on the selected studies, check references by looking at works of authors already included, since they obviously carry out relevant research in relation to their objectives; 3. Based on the set of documents found, check studies that cite the selected studies (forward snowballing). It is recommended to use Google Scholar as it captures more than individual databases.

3.4. Inclusion and Exclusion Criteria

With the intention of selecting only the most relevant articles for the study object, inclusion and exclusion criteria were applied. For the selection of the articles, the following inclusion criteria were defined: 1. Articles that present measurement tools, methods, models, or approaches based on measurement used in Software Engineering and articles that present a discussion on measurement tools used in the software development process; 2. Articles that contain title, abstract, and keywords related to the research questions; 3. Articles published between 2007 and 2018, to get tools relatively recent and more likely to be active.

The exclusion criteria applied in this study were: 1. Duplicate studies. When a study has been published in more than one conference, workshop or journal, the most complete version will be used, i.e., the one which explains it in more detail; 2. Studies outside the scope of this research, that not present measurement tools; 3. Incomplete Articles (published as Short Paper, less than 4 pages).

3.5. Systematic Literature Review Limitations

This systematic literature review (RSL) presented in this paper has some limitations and weaknesses. We describe as follows the threats to validity of this RSL, as well as mitigation strategies for each one: **Research questions**: the defined questions might not have covered the whole measurement and metrics tool area. Thus, one may not find answers to the questions that concern him/her. As we considered this a feasible threat, several discussion meetings with the research team were held to calibrate the RSL questions; **Subjectivity in the study selection**: we cannot guarantee that all relevant primary studies were selected. It is possible that relevant papers were not chosen. In order to mitigate it, we performed the automatic search, and complemented it by performing manual search to try to collect all primary studies in this field; **Subjectivity in the data extraction**: during the data extraction process, the primary studies were classified based on our judgment. In order to mitigate this threat, the classification process was performed using peer review; and **Repeatability of the systematic process**: there is a risk involving the ability to replicate or extend this study. This threat is mitigated through a detailed description of the systematic process in this work, since all details of the study protocol were described. Moreover, we published the data extraction results on the Web as an additional source of information.

3.6. Data Collection and Analysis

After obtaining the primary studies that met the established inclusion and exclusion criteria, the collection and analysis of data was performed. The Start tool [2] was used to organize the information from the articles that came up from the search, as this tool assists in the systematic review process. The results that the search string returned during its execution in the research bases were grouped in this tool for the selection of the publications according to the inclusion and exclusion criteria and for the process of data extraction.

The first stage in the process of selecting the primary studies consisted of the execution of the search strategy presented in Section 3.3, that is, inserting the strings in the research base chosen. With this stage, 215 articles were found, The majority of them, 81% were obtained through the Scopus digital library, while 19% of the articles came from the Science Direct library.

The second stage of data collection consisted in reading the title, abstract, and keywords of the 215 articles to verify if the article was within the inclusion and exclusion criteria defined. Of the 215 articles found in stage 1.78% were rejected, 19% were accepted, and 3% were rejected for being duplicates. The great rejection ratio was mainly due to the fact that the articles didn't meet the "Articles published between 2007 and 2018" inclusion criteria.

[2] http://lapes.dc.ufscar.br/tools/start_tool

In the third stage of the data collection, a complete reading of the articles selected in the second stage was done. This way, the article was evaluated to make sure it could contribute in answering the research question. Besides that, the other exclusion criteria were also applied. After the complete reading of the articles, 59%, or 24 articles, were accepted, meaning they answered the research question and did not present the other exclusion criteria.

During data extraction, the researchers carefully read the primary studies. The peer-review process was put in place and two researchers extracted data for the same study. Disagreements were solved by a third researcher. We performed a pilot of the data extraction, aiming to align the understanding of the researchers for answering the research questions. The pilot was performed with five randomly chosen primary studies, and the researchers discussed about the disagreements on the individual answers.

In summary, in the first stage, where the search string was inserted in the digital libraries, a total of 215 articles were obtained. In the second stage, the results were filtered by reading their title, abstract, and keywords, and a total of 41 articles were selected. In the third stage, another filter was applied, through the complete reading of the article, with the object of finding the answers to the research questions. After this stage, 24 articles had been chosen to answer the research questions.

In addition to the **24 articles** selected in the automatic search, 3 other articles were selected through manual search. These articles were not returned in the automatic search with the search string, however they were found by manual search and considered relevant to be inserted in this study. After finishing the 3 analysis stages, **27 articles** had been selected for data extraction. Table 1 presents the list of articles selected for that, with its respective title, author, and year of publishing.

As per the inclusion criteria, only articles published from 2007 onwards were included. As such, three articles related to measurement tools were published in 2007, five articles in 2008, two articles in 2009, two articles in 2010, three articles in 2011, three articles in 2012, two articles in 2013, no articles in 2014, three articles in 2015, two articles in 2016, and two articles in 2017. It's possible to see that the year of 2008 was the one that presented the highest scientific contribution in the measurement tools field, since five articles were published that year. 2014 is also noteworthy, as it had no publications on the subject.

4. Results Systematic Literature Review

4.1. RQ.1: Tools

Many different tools that perform the collection of measurements were found. Some tools shown by the articles have been discontinued, that is, they are no longer available for download or use. Therefore, for each tool found through the systematic review, the availability of the tool was verified by checking if the article had the link for access and searches ere download, and also done on Google. The research questions considered only the available tools. Table 2 shows the tools that are available for download and use, and the article'(s) reference them.

Some of these tools perform the collection of metrics of the product, while others collect metrics of the product and process. Table 3 presents the name of the tool and the Measurement Type it collects.

4.2. RQ.2: Functionalities

The measurement tools presented by the selected articles have different functionalities and characteristics. For instance, many tools support a single programming language, while others support many of them. The main characteristics of the tools found in the literature review are: Graphics (G); Integration (I); Multiple Languages (M); Definition of Metrics (D);Reports (R); Export (R); Import(Im). Table 4 presents the main functionalities of the tools.

The **graphics** functionality means that the tool allows for the creation of graphics for metrics analysis. The **integration** functionality informs that the tool performs integration with other tools. **Multiple languages** implies that the tool performs the collection of more than one programming language. The **metrics definition** means that the tool allows the user to create their own metrics. The **report** functionality means that the tool generates some kind of report for metrics analysis. The **export** functionality informs that the tool allows for the exportation of its data. And the **import** functionality represents that the tool allows for the importation of data.

4.3. RQ.3: Metrics

The measurement tools identified in the systematic literature review relate/use different metrics. Table 5 showcases the main metrics presented by the tools, as well as the occurrence of each of the metrics and the tools that use the respective metric.

Table 1. Articles Selected for Data Extraction

ID	Paper Title	Author	Year
S1	A metrics tool for multi-language software	[14]	2007
S2	Design optimization metrics for UML based object-oriented systems	[15]	2007
S3	Managing software process measurement: A metamodel-based approach	[16]	2007
S4	A metamodel for the measurement of object-oriented systems: An analysis using alloy	[17]	2008
S5	How to measure agile software development	[18]	2008
S6	Software metrics for agile software development	[19]	2008
S7	”Unit metrics” - A tool to support refactoring in agile software development	[20]	2008
S8	Using a combination of measurement tools to extract metrics from open source projects	[21]	2008
S9	A coupling and cohesion metrics suite for object-oriented software	[22]	2009
S10	Analysis and implementation of software metric for object-oriented	[23]	2009
S11	A framework for source code metrics	[24]	2010
S12	Towards a 'Universal' software metrics tool: Motivation, process and a prototype	[25]	2010
S13	A pluggable tool for measuring software metrics from source code	[26]	2011
S14	SMIILE prototype	[27]	2011
S15	XML-based integration of the SMIILE tool prototype and software metrics repository	[28]	2011
S16	Customizing GQM models for software project monitoring	[29]	2012
S17	Towards the better software metrics tool	[30]	2012
S18	Validation of measurement tools to extract metrics from open source projects	[31]	2012
S19	A Methodology for Obtaining Universal Software Code Metrics	[32]	2013
S20	ASSIST: An integrated measurement tool	[33]	2013
S21	An object based software tool for software measurement	[34]	2015
S22	Evaluating metrics at class and method level for java programs using knowledge based systems	[35]	2015
S23	Integration of software measurement supporting tools: A mapping study	[36]	2015
S24	A systematic literature review on software measurement programs	[37]	2016
S25	Building a user oriented application for generic source code metrics extraction from a metrics framework	[32]	2016
S26	An ontology-based approach for integrating tools supporting the software measurement process	[38]	2017
S27	Investigating differences and commonalities of software metric tools	[39]	2017

5. Discussion of results

5.1. Tools

Many different tools that collect metrics on the Software Engineering field were found. The focus of these tools is in product or the process. Table 5 presents the distribution of the tools that were found during the systematic review. Of the 22 tools found during the systematic review, 19 of them, around 86%, are tools that collect only measurements of product, or code metrics. No tool provides collection of process metrics exclusively, and four other tools perform the collection of both product and process metrics. Figure 1 shows tool distribution for each measurement type.

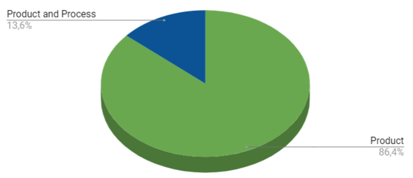

Figure 1. Distribution of Tools in Regards to Metrics Provided.

To evaluate the tool relevance, each one of the articles that cites a tool was researched in Google Scholar. Google Scholar returns the number of times that the article was cited. Then a summation of the amount of citation of all articles for a specific tool was made. Table 6 shows the relevance of the tools.

The UnitMetric tool proved itself the most relevant among the set of tools found in the systematic review, as it was cited 44 times. The CCMETRICS tool took the second place in relevance, being cited 22 times. The MASU tool stayed in third place, being mentioned 21

Table 2. Available Tools

Tool Name	Articles
CCCC	[39]
CCMETRICS	[22]
Chidamber and Kemerer Java Metrics	[21] [31]
Code Analyzer	[39]
CMT++	[40]
CMTJAVA	[40]
DePress	[38] [36]
Eclipse Metrics Plugin 1.3.6	[39]
EPM	[29]
Essential metrics	[40]
JArchitect	[39]
JAM	[35]
JHawk	[31]
JMetric	[40]
LocMetrics	[39]
MASU	[26]
RSM	[40] [21] [31]
SAS	[37]
Source Monitor	[39]
Stan4j	[39]
Understand	[39]
UnitMetrics	[19] [20] [18]

Table 3. Measuring Tools

Measurement Type	Tools
Product	CCCC, CCMETRICS, Chidamber and Kemerer Java Metrics, Code Analyzer, CMT++, CMTJAVA, Eclipse Metrics Plugin 1.3.6,Essential metrics, JArchitect, JAM, JHawk, JMetric, LocMetrics, MASU, RSM, Source Monitor, Stan4j, Understand, UnitMetrics
Product and Process	DePress, EPM, SAS

times.

5.2. Functionalities

Of the **22 tools analyzed**, twelve tools, or 54% of them, have support for reports. Ten tools, or 45% of them, have support for multiple programming languages. Nine tools, or 41% of them, perform the creation of graphs. Six tools perform data exporting.

Table 4. Main Functionalities

Tool	G	I	M	D	R	E	Im
CCCC			x			x	
CCMetrics	x		x				
Chidamber and Kemerer Java Metrics					x	x	
Code Analyzer			x		x		
CMT++			x		x		
CMTJAVA					x		
DePress		x			x	x	x
Eclipse Metrics Plugin 1.3.6						x	
EPM	x			x			x
Essential metrics			x		x		
JArchitect		x			x		
JAM	x						
JHawk	x			x	x		
JMetric					x		
LocMetrics	x		x				
MASU			x				
RSM			x		x		
SAS					x		
Source Monitor	x		x			x	
Stan4j	x				x		
Understand	x		x	x			
UnitMetrics	x					x	

Three tools allow for the definition of metrics. Finally, two tools performs integration with other tools, and two tool performs data importing.

This shows that the focus of the tools is to **generate reports**. Besides that, there is interest in supporting multiple programming languages and generating graphs to assist in metrics analysis.

5.3. Identified Metrics

Among the tools analyzed, the **lines of code** metric is the most relevant one, appearing in **12 tools** out of the 22 total. The **cyclomatic complexity** metric appears in **11 tools** and the **coupling between objects** one appears in **11** of them. These are the three main metrics presented by the selected tools.

The tools selected in the primary studies present the use of other metrics, but all of them have a ratio of appearance lower than 3%, and as such were dismissed from the study, as they are mentioned in very few articles. The definitions of the main metrics used by the tools are presented below:

Table 5. Main Metrics Presented by the Tools

Metric	Frequency	Tools
Lines of Code (LOC)	40%	CCCC, CMT++, CMTJAVA, Code Analyzer, Eclipse Metrics Plugin, JAM, LOCMetric, RSM, Source Monitor, Stan4J, Understand, UnitMetrics
Cyclomatic Complexity (CC)	36,6%	CCCC, CMT++, CMTJAVA, Eclipse Metrics Plugin, Essential Metrics, JAM, JArchtect, JMetric, MASU, Stan4J, Understand
Coupling Between Objects (CBO)	36,6%	CCCC, Chidamber and Kemerer Java Metrics, Essential Metrics, JAM, JArchtect, JHawk, MASU, RSM, Stan4J, Understand, UnitMetrics
Depth of Inheritance Tree (DIT)	30%	CCCC, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JArchtect, MASU,RSM, Stan4J, Understand
Number of Children (NOC)	30%	CCCC, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JArchtect, MASU, RSM, Stan4J, UnitMetrics
Lack of Cohesion in Methods (LCOM)	23,3%	Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JArchtect, MASU, Stan4J, Understand
Weighted Methods per Classe (WMC)	23,3%	CCCC, Chidamber and Kemerer Java Metrics, Eclipse Metrics Plugin, JAM, JHawk, MASU, Stan4J
Responde for Class (RFC)	13,3%	Chidamber and Kemerer Java Metrics, JAM, MASU, Stan4J

Table 6. Relevance of Tools

Tool Name	Relevance
UnitMetrics	44
CCMETRICS	22
MASU	21
SAS	10
DePress	7
RSM	6
Chidamber and Kemerer Java Metrics	5
CMT++	1
CMTJAVA	1
CCCC	0
Code Analyzer	0
Eclipse Metrics Plugin 1.3.6	0
EPM	7
Essential metrics	1
JArchitect	0
JAM	0
JHawk	5
JMetric	1
LocMetrics	0
Source Monitor	0
Stan4j	0
Understand	0

5.3.1. Lines of code The Lines of Code (LOC) metric calculates the size of the software, measuring the quantity of lines of code that it has [35].

One way of calculating this metric is presented by [4]. This study presents two metrics to perform the calculation, the Non-commented Line of Code (NCLOC) metric and the Commented Lines of Program Text (CLOC) metric. As such, to calculate the line of code metric, the following equation is used:

$$LOC = NCLOC + CLOC \tag{1}$$

This measurement is in absolute scale.

5.3.2. Cyclomatic Complexity The cyclomatic complexity metric measures the number of linearly independent paths that the software has [4]. To calculate the cyclomatic complexity in a software with an F fluxogram, the following equation is used:

$$v(F) = e - n + 2 \tag{2}$$

Where *e* represents the edges and *n* the nodes.

The higher the cyclomatic complexity of a system, the harder it's maintenance will be [4].

5.3.3. Coupling Between Objects The Coupling Between Objects (CBO) metric calculates the degree of dependency among modules [4].

An example of how to calculate this metric is presented by [41]. In this study, we show how to calculate coupling between two modules. To calculate coupling between these modules, you first need to classify the types of relations existing between them. These relations can be [41]:

- **R5**: If X ramifies from Y, alters data or declarations in Y;
- **R4**: If X and Y reference the same global data;
- **R3**: If X sends parameters to Y seeking to control its behavior;
- **R2**: If X and Y accept the same type of registry as parameter;
- **R1**: If X and Y communicate through parameters;
- **R0**: If X and Y do not communicate.

The equation presented is:

$$M(X,Y) = i + \frac{n}{n+1} \tag{3}$$

Where **i** represents the highest type of relation, and **n** represents the number of interconnections between **X** and **Y** [41]. This measurement is in ordinal scale.

5.3.4. Depth of Inheritance Tree The Depth of Inheritance Tree (DIT) metric calculates the depth of the class in the project's hierarchy. The more depth a class has, the harder it will be to foresee its behavior [35], [26].

5.3.5. Number of Children The Number of Children (NOC) metric calculates the amount of classes derived from the class being measured [26]. The calculation of this metric is done by adding the immediate children of the class [4].

5.3.6. Lack of Cohesion in Methods The Lack of Cohesion in Methods (LCOM) metric measures the discrepancy of methods in a class when faced with instanced variables [35]. A way of calculating this metric is presented by [26]. The calculation is done by taking each pair of methods within the same class, verifying if they possess disjoined access to a set of instance variables. If so, increment P by 1. Otherwise, if they have at least one variable in common, increase P in one. This way:

$$LCOM = \begin{cases} P - Q & \text{if P >Q} \\ 0 & \text{if else} \end{cases} \tag{4}$$

5.3.7. Weighted Methods per Class The Weighted Methods per Class (WMC) metric performs the sum of the complexities of the methods in the class [26]. The calculation of this metric is done through the sum of each weighted method according to its complexity [4]. Therefore, we have the following equation:

$$WMC = \sum_{i=1}^{n} c_i \tag{5}$$

5.3.8. Response for Class The Response for Class (RFC) metric represents the size of a set of answers for the class. This set of answers consists of all the methods called by local methods [4]. The calculation of this metric is done through the sum of the number of local methods plus the number of methods called by the local methods [4].

6. Conclusion

The use of metrics assist software development organizations in making trustworthy estimates in regards to the software product being developed. To facilitate the measurement process, the use of tools is indicated to minimize possible mistakes.

Literature presents many tools to perform the collection of metrics. The tools collect, mainly, product metrics. Besides that, the tools have different functionalities. Most tools present the functionalities of supporting multiple programming languages, and generating graphs. It's possible to observe a tendency in the metrics used by measurement tools, showing a pattern in the most commonly used metrics.

This study achieved its proposed objective, since the analysis of the tools allowed for a higher understanding of their functionalities and of the metrics collected by them. As such, organizations can use this information as a base to choose a measurement tool that best fits their business.

As a future study, the snowballing technique will be applied to ensure that all relevant articles were covered. A grey literature review will be made to know market practices and which tools are being used. Besides that the information gathered in this study will be used to identify gaps in existing measurement tools. This will lead to the proposal of a measurement tool that fills these gaps and uses the main metrics listed in this study.

7. Acknowledgments

This research work has the support of the Research Support Foundation of the Federal District (FAPDF).

References

[1] A. S. Nuñez-Varela, H. G. Pérez-Gonzalez, F. E. Martínez-Perez, and C. Soubervielle-Montalvo, "Source code metrics: A systematic mapping study," *Journal of Systems and Software*, vol. 128, pp. 164–197, 2017.

[2] W. A. Florac, R. E. Park, and A. D. Carleton, *Practical Software Measurement: Measuring for Process Management and Improvement*. Pittsburgh, PA 15213: Carnegie Mellon University, 1997.

[3] H. Kinnunen and E. Luoma, "Towards measuring the agility of software business," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[4] N. Fenton and S. L. Pfleeger, *Software Metrics (2Nd Ed.): A Rigorous and Practical Approach*. Boston, MA, USA: PWS Publishing Co., 1997.

[5] S. H. Kan, *Metrics and Models in Software Quality Engineering*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2nd ed., 2002.

[6] J. E. Cook and A. L. Wolf, "Software process validation: quantitatively measuring the correspondence of a process to a model," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 8, no. 2, pp. 147–176, 1999.

[7] L. Finkelstein and M. Leaning, "A review of the fundamental concepts of measurement," *Measurement*, vol. 2, no. 1, pp. 25–34, 1984.

[8] D. Galin, *Software Quality: Concepts and Practice*. John Wiley & Sons, 2018.

[9] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *arXiv preprint arXiv:1801.06475*, 2018.

[10] ISO/IEC/IEEE, "Systems and software engineering - measurement process," standard, International Organization for Standardization, 2017.

[11] N. Fenton, "Software measurement: A necessary scientific basis," *IEEE Transactions on software engineering*, vol. 20, no. 3, pp. 199–206, 1994.

[12] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *ech. rep. EBSE 2007-001*, 2007.

[13] C. Wohlin and R. Prikladniki, "Systematic literature reviews in software engineering," *Information and Software Technology*, vol. 55, no. 6, pp. 919–920, 2013.

[14] P. Linos, W. Lucas, S. Myers, and E. Maier, "A metrics tool for multi-language software," in *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications*, pp. 324–329, ACTA Press, 2007.

[15] E. Ramaraj and S. Duraisamy, "Design optimization metrics for uml based object-oriented systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 03, pp. 423–448, 2007.

[16] F. Garcia, M. Serrano, J. Cruz-Lemus, F. Ruiz, M. Piattini, A. R. Group, *et al.*, "Managing software process measurement: A metamodel-based approach," *Information Sciences*, vol. 177, no. 12, pp. 2570–2586, 2007.

[17] J. McQuillan and J. Power, "A metamodel for the measurement of object-oriented systems: An analysis using alloy," in *Software Testing, Verification, and Validation, 2008 1st International Conference on*, pp. 288–297, IEEE, 2008.

[18] M. Kunz, R. R. Dumke, and A. Schmietendorf, "How to measure agile software development," in *Software Process and Product Measurement*, pp. 95–101, Springer, 2008.

[19] M. Kunz, R. R. Dumke, and N. Zenker, "Software metrics for agile software development," in *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*, pp. 673–678, IEEE, 2008.

[20] M. Kunz, N. Zenker, S. Mencke, and R. R. Dumke, "Unit metrics-a tool to support refactoring in agile software development.," in *Software Engineering Research and Practice*, pp. 389–395, 2008.

[21] N. Awang Abu Bakar, C. V. Boughton, *et al.*, "Using a combination of measurement tools to extract metrics from open source projects," in *Proceeding (632) Software Engineering and Applications-2008*, ACTA Press, 2008.

[22] S. Husein and A. Oxley, "A coupling and cohesion metrics suite for object-oriented software," in *Computer Technology and Development, 2009. ICCTD'09. International Conference on*, vol. 1, pp. 421–425, IEEE, 2009.

[23] E. Da-wei, “Analysis and implementation of software metric for object-oriented,” in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pp. 1–4, IEEE, 2009.

[24] N. Maneva, N. Grozev, and D. Lilov, “A framework for source code metrics,” in *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, pp. 113–118, ACM, 2010.

[25] G. Raki, Z. Budimac, and K. Bothe, “Towards a 'universal' software metrics tool: Motivation, process and a prototype,” vol. 2, pp. 263–266, 2010. cited By 1.

[26] Y. Higo, A. Saitoh, G. Yamada, T. Miyake, S. Kusumoto, and K. Inoue, “A pluggable tool for measuring software metrics from source code,” in *Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pp. 3–12, IEEE, 2011.

[27] G. Rakić and Z. Budimac, “Smiile prototype,” in *AIP Conference Proceedings*, vol. 1389, pp. 853–856, AIP, 2011.

[28] G. Rakić, Č. Gerlec, J. Novak, and Z. Budimac, “Xml-based integration of the smiile tool prototype and software metrics repository,” in *AIP Conference Proceedings*, vol. 1389, pp. 869–872, AIP, 2011.

[29] A. Monden, T. Matsumura, M. Barker, K. Torii, and V. R. Basili, “Customizing gqm models for software project monitoring,” *IEICE TRANSACTIONS on Information and Systems*, vol. 95, no. 9, pp. 2169–2182, 2012.

[30] Z. Budimac, G. Rakic, M. Hericko, and C. Gerlec, “Towards the better software metrics tool,” in *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 491–494, IEEE, 2012.

[31] N. S. A. A. Bakar and C. V. Boughton, “Validation of measurement tools to extract metrics from open source projects,” in *Open Systems (ICOS), 2012 IEEE Conference on*, pp. 1–6, IEEE, 2012.

[32] A. S. Núñez-Varela, H. G. Pérez-González, F. E. Martínez-Pérez, and J. Cuevas-Tello, “Building a user oriented application for generic source code metrics extraction from a metrics framework,” in *Software Engineering Research and Innovation (CONISOFT), 2016 4th International Conference in*, pp. 27–32, IEEE, 2016.

[33] B. Keser, T. Iyidogan, and B. Ozkan, “Assist: an integrated measurement tool,” in *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*, pp. 237–242, IEEE, 2013.

[34] G. Santhoshini and K. Anbazhagan, “An object based software tool for software measurement,” in *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pp. 1–5, IEEE, 2014.

[35] E. Umamaheswari, B. Natarajan, and D. Ghosh, “Evaluating metrics at class and method level for java programs using knowledge based systems,” vol. 10, pp. 2047–2052, 01 2015.

[36] V. S. Fonseca, M. P. Barcellos, and R. de Almeida Falbo, “Integration of software measurement supporting tools: A mapping study.,” in *SEKE*, pp. 516–521, 2015.

[37] T. Tahir, G. Rasool, and C. Gencel, “A systematic literature review on software measurement programs,” *Information and Software Technology*, vol. 73, pp. 101–121, 2016.

[38] V. S. Fonseca, M. P. Barcellos, and R. de Almeida Falbo, “An ontology-based approach for integrating tools supporting the software measurement process,” *Science of Computer Programming*, vol. 135, pp. 20–44, 2017.

[39] L. Aversano, C. Grasso, P. Grasso, and M. Tortorella, “Investigating differences and commonalities of software metric tools,” pp. 249–256, 01 2017.

[40] A. Núñez-Varela, H. G. Perez-Gonzalez, J. C. Cuevas-Tello, and C. Soubervielle-Montalvo, “A methodology for obtaining universal software code metrics,” *Procedia Technology*, vol. 7, pp. 336–343, 2013.

[41] N. Fenton and A. Melton, “Deriving structurally based software measures,” *Journal of Systems and Software*, vol. 12, no. 3, pp. 177–187, 1990.