ISSN 2347 - 3983
Volume 7, No. 9 September 2019
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter01792019.pdf
https://doi.org/10.30534/ijeter/2019/01792019

An efficient Software Source Code Metrics for Implementing for Software quality analysis

Varun K L Srivastava[1], N. Chandra Sekhar Reddy[2], Dr. Anubha Shrivastava[3]

[1]Director General , Department of R & D, Association of Education Training & Research Institute(ASSOED), Noida (UP) 201301, INDIA. mr.varunsrivastava@gmail.com
[2]Department of Computer Science and Engineering, MLR Institute of Technology, Dundigal, Hyderabad, 500043. India. nreddy28208@gmail.com
[3]Associate Professor, G. L. Bajaj Institute of Technology and Management, Noida (UP) 201301 , India Aanubha.shrivastava@glbitm.org

ABSTRACT

The challenging task is evaluating size of a complex and enormous software framework. Primarily in life cycle of project, while necessities for framework might be functional & immature described only at extreme level, profiles of resource are require for suitable staffing, funding, and progress of feasible project strategy. Same project historical software size information& trends gives a device to calculate size of software, making a possible evaluation method. As processors are being utilized in each and every imaginable region in current world, software quality gets a main feature in planned achievement of a human & business safety in common. Discovering quality factors of software & illustration those into computable measures will be an essential feature in viable achievement of end product. Program features illustration into these values of metrics depicts information framework behavior & structural complexity. In this survey, 5 software metric are utilized, they are lines of code (LOC), lines of comment (COM), cyclomatic complexity (MVG), number of modules (NOM), and Halstead volume (HV) have been used to examine a group of 3 sorting programs of java. The 3 software measurement devices have been applied on them to confirm their presentation w.r.to metrics cited there also a resultant metric maintainability index has been measured from basic metrics to designate comparative maintainability of the source code. The comparative study of selected devices has been undertaken to expose how they change in providing outcomes for similar programs. Additionally, few other quality features that might be resultant from essential metrics are cited in next sub section.

Key words: Software Metrics, Software Quality, Software Testing, Software Faults, Software Engineering

1. INTRODUCTION

Why Evaluate Software Quality: Suppose you get a programming item that is conveyed looking into time, inside budget, and effectively & proficiently perform know its specified works.

Does it follow that you will be happy with it? For a few reasons, the response might a chance to be no. Here are a few of the basic issues you might find: 1. the programming item might a chance to be hard should see Furthermore Was troublesome to change. This prompts unreasonable expenses in programming maintenance, Also these fetches would not insignificant. To example, a later paper toward Elshoff [1] demonstrates that 75% for general Motors' programming exert is used in product maintenance, also that GM may be honestly commonplace for substantial industry programming exercises. 2. The programming item might be troublesome to utilize alternately simple should abuse. A late GAO report card [2] distinguished through $10,000,000 over unnecessary legislature expenses because of ADP issues; a large number about them were a direct result those product might have been in this way simple to abuse. 3. The programming item might make unnecessarily machined dependent or tough will coordinate with other projects. This issue is troublesome sufficient now, be that as machine sorts proceed to proliferate; it will get more awful. Major product personal satisfaction choice focuses. There are an amount from claiming commonplace circumstances for which it will be time permits should push a solid impact with respect to programming quality, and for which it will be critical will need a great seeing of the different aspects of product

caliber. Here are a few: 1. Get ready the caliber determinations for a programming item. Planning the thing that capacities you need what's more entryway significantly execution (speed, accuracy) you need aid equitably clear. Demonstrating that you also require on look after capability alternately understandability may be important, be that considerably all the more was troublesome on define on a few testable design. 2. Checking for consistence for personal satisfaction determinations. This may be vital Assuming that the personal satisfaction determinations are should make serious. It could obviously make done with an extensive financing of great people, at this sort checking is both unreasonable and diligent once human's spirit. 3. Making fitting outline trade-offs between improvement expenses and operational expenses. This may be particularly vital a result tight improvement plans alternately schedules make activities on hold back looking into maintainability, portability, & usability. 4. Programming bundle determination. Here again, numerous clients necessity An relative evaluation from claiming how effectively every one bundle might be adjusted will their installation's evolving necessities and equipment build programming building is equitably educated support and urgent configuration procedure due to today's progressive surroundings which may be truly capricious also in principle, not fully specifiable ahead of time. Compelling product caliber assessment obliges determinants that describe what nature will be and more entryway it can be followed over of the improvement methodology or the finished item itself. Product industry may be bit by bit progressing towards a time from claiming high maturity; the place casual methodologies will personal satisfaction Investigation can never again fill. Because of the revolutionary growth, clients need aid likewise distinguishing its quality and they need aid not eager to bargain on the qualitative parts. Notwithstanding from claiming all this, inward caliber of an item might try unchecked alternately make deliberately compromised now and again. Product measurements are primitive indicators on code caliber that provide us with the methods should make pro-active activities at most punctual phase possible, at whatever point one task will be moving off-track.

Quality need separate elucidation for distinctive individuals. Different personal satisfaction guidelines exist which need aid pertinent to the associations included for programming advancement. ISO and IEEE would those practically generally utilized norms in this field. ISO/IEC 9126 [8] characterizes reliability functionality, maintainability, efficiency, usability& portability as nature aspects for product results. IEEE need distributed a standard to those product nature measurements technique [9]. IEEE characterizes quality of software- degree should a component, system, or procedure meets specified necessities or client desires. Further, programming measurements need aid instruments connected with a bit for programming or its configuration determinations for those objective will accomplish proliferation quantitative measurements, which might make further connected on expense estimation, undertaking scheduling, debugging, personal satisfaction certification and indistinguishable.

2. BACKGROUND

Estimation will be in any building domain; also there is no exception on product building. A few specialists in the secret word bring connected product measurements as enter inputs with aide personal satisfaction predictors. The work [10] identifies connection between a few measurements starting with well-known article situated measurements suites for example, CK metrics, McCabe Cyclomatic multifaceted nature Also Different measure metrics, also showing could reasonably be expected thresholds. The work [6] recommend new product measurements In view of coding measures violations will catch idle faults in a advancement. The work [5] identifies a straight development pattern in product size for crewed space Also aircraft, which might sensibly anticipate product extent in comparative future programs, utilizing SLOC built information. The work [7] researches the connections of size & unpredictability measurements for maintainability of open source software. The work [7] uses CK metrics, SLOC, COM measurements and so on. On examine the association the middle of product measurements Furthermore defects.

3. METHODOLOGYFOLLOWED

There would 2 methodologies to programming estimation. Particular case may be centered around regulate assessment of the personal satisfaction about finished item handled Throughout Different processes; Furthermore in the second one, procedures themselves need aid measured should illuminate for duration, cost, adequacy & effectiveness of programming advancement exercises. In this survey, we proposed with assess source code as finished item to metric built dissection. On start with, projects need aid chose for which measurements should be observationally approved to. We have opted to three java based foray projects starting with well-established calculations of air pocket sort, determination kind Furthermore fast kind. Afterward a suitableness situated of measurements from claiming investment is picked. This in place obliges determination and pre-testing for devices which would dialect compatible,

backing provided for measurements and on the foundation about accessibility. After actualizing the devices and catching metric values, an inferred metric Maintainability list (MI) may be ascertained from build metrics; outcomes need aid compared Also translated inevitably. Figure 1 demonstrates the technique took after in this manuscript:

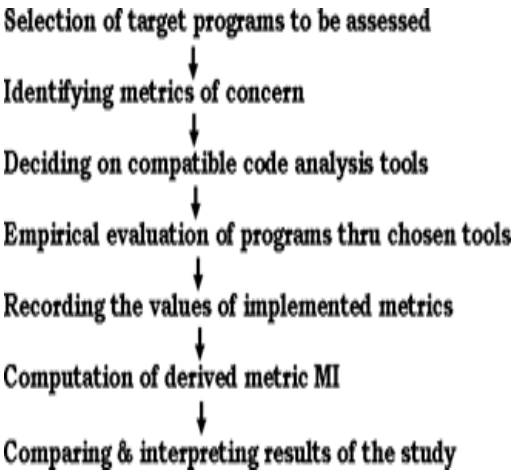

Figure.1: Methodology Followed

There exist numerous open-source & business estimation devices with look over depending upon those investigator inclination and other similarity problems. In this manuscript, devices supporting examination from claiming java projects were needed. After exactly preliminary examination, three instruments have been chose – source monitor, C & C++ code counter, and JHawk. For the purpose for concision, they would identifier as SM, CCCC, & JHK separately from this side of the point. One cause behind opting to various instruments is to place crosswise over the contrasts also likenesses prevailing around them done delivering effects. Instruments process a number measurements qualities crazy from claiming which outcomes of five measurements about premium need aid recorded, also these are: NOM, LOC, MVG, COM, and HV. Out about these 5 essential metrics, 3 bring further been used with figure MI similarly as work for LOC, HV, & MVG. Some other derivable quality components need aid summed- upon.

4. METRICS UNDER CONSIDERATION

A perfect consideration of the classification for code qualities & possibility for their provision in enhancing result of potential activities prompted to a research body mainly joining acceptance about these measurements. Further these need aid fit for decreasing subjectivity throughout quality promise & aides in choice making because of their way about reproducible. There exist a few regulate and backhanded measures, crazy of which five measurements have been opted for those devices to make inspected. Ahead may be a short portrayal about them.

4.1 Line of Count (LOC) – Physical Size

This is much prevalent size-oriented metric displays entire number of non-comment lines, non-blank. Supporters of the LOC measure case that LOC may be an "artifact" for every one product improvement tasks that might make effectively counted, that a significant number existing product estimation models utilize LOC or KLOC as main information with assess different viewpoints of quality & cost [11].

4.2 NOM (Number of modules) –Code Distribution

All functions, methods are count below this physical and in addition legitimate metric. As contrasted with LOC, it is a greater amount serious a size-metric on account of with a percentage extent, it is autonomous of the modifying dialect opted for. It may be simple to ascertain and serves best likewise an interface metric. The greater amount modules a population have additional perplexing its interface will be expected to be.

4.3 COM (Number of Lines of Comments)-Documentation

Well-documented programming helps maintainers &developers just as great. COM speaks to the downright sourball remark number and further as a trait of the understandability, maintainability, &measures-reusability. An additional advantageous metric called CCR (Code with remark Ratio) might make inferred starting with this measure with have an evaluate for upon what amount of the source code may be great recorded.

4.4 Halstead Volume (HV)

HV is a measure from crew of Halstead metrics, may be a composite metric In view of number about (distinct) operands& operators in source code. As stated by Halstead Volume may be those number of number of mental correlations required will produce a system. It will be computed likewise the system period times the 2-base logarithm of vocabulary extent. It speaks to those volume for majority of the data (in bits) obliged with define a project. HV depicts text based code intricacy and will be a standout amongst the important parameter in registering maintainability list.

5. TOOLS DESCRIPTION

5.1 C and C# Code Counter (CCCC)

CCCC might have been formed to 2001 by Tim minimal reasonable as a fragment of this doctorate examination venture. It will be free-ware open wellspring order line

Interface initially intended to Linux, as well as build-able

on the Win32 stage. Initially actualized will methodology C# & ANSI C programs, resulting variants have the capacity with transform source files of java as well. It will be not difficult to run on order line by specifying names one or more source files to be investigated. CCCC will primary check the development of file name & assuming that the development may be recognized as demonstrating a backed language, fitting parser will run on record. As every record will be parsed, ID number of specific constructs will reason records should make composed under an interior database. Last yield will a chance to be created in XML files& HTML arrangement. The CCCC creates different measures like that extent metrics, unpredictability metrics, and object turned measurements from CK and some others.

5.2 Source Code Monitor (SM)

Enhanced by the programming with graphical-interface, sourball screen [12] will be a free-ware closed-source programming estimation device. It may be fit will be worked looking into ASCII content files made ahead different frameworks anyway runs only on Windows. The "check pointing" may be a standout amongst its different characteristics to keep the outcomes around thereabouts that manager of projects could perceive how project code progressions over time. There would five diverse perspectives accessible should show the effects like charts view, checkpoint view, details view, project view, and method view. The languages maintained are - VB. NET, C, VB6, HTML, C#, C++, Java, and a couple others. You quit offering on that one might send out resultant measurements information from sourball screen to quick files, XML or CSV design. Measurements help differ slightly with selection of programming language, nevertheless most usually caught ones are- techniques per class, percent branch statements, LOC, maximum method complexity, classes & interfaces, & percent lines with the comments.

Principally a JHawk, java metric tool [13] need advanced from a stand-alone GUI provision to incorporate an order transport form and an eclipse plug in. It compromises to process IDE coordination (for Visual period for Java) Also gives the HTML, XML, and CSV send out formats. Separated from letterset printing those clients make their novel metrics, it gives a dashboard tab that provides for a fast review of the measurements during System, one bundle and population level. Also, the JHawk information viewer permits a client with perspective progressions to center measurements about whether – for case through an extend lifecycle.

6. EXPERIMENTAL EVALUATION

The analysis of code was executed after this preliminary study & pre-preparations. The 3 programs of java dependent upon 3 categorization strategies – Selection sort, Quick sort, & Bubble sort for were investigated through the devices embraced. Short portrayal of the source projects is in table I.

Table 1: Short portrayal of the source projects

Symbolic names of programs	Explanation
ProgramX	Bubble sort
ProgramY	Selection sort
ProgramZ	Quick sort

Every project may be accessed through every last one of three devices so that comes about might be compared crosswise over different instruments. As stated by the distinctive tool's metric support, various measurements values were estimated &conveyed naturally as part of outcomes. Though just the measurements of concentration were caught & recorded in table 2 for further examination.

Table 2: Results of tools' implementation

Tools	Prog	MVG	COM	HV	LOC	NOM
CCCC	ProgA	5	1		57	3*
	ProgB	5	2	-	30	2*
	ProgC	11	3		45	2*
Source Code	ProgA	4	$1^{\#}$		44	4
Monitor	ProgB	4	$2^{\#}$	-	32	2
(ScM)	ProgC	9	$3^{\#}$		40	2
JHawk	ProgA	5	1	318.0	47	4
	ProgB	4	2	519.7	36	2
(JHK)	ProgC	8	3	727.3	42	2

- indicates metric is not supported by corresponding tool
# indicates normalized values according to Table III (row 4, col2)
* indicates different granularity level according to Table III (row3, col2)

It will be clear in table II that to the similar program, indistinguishable measurements prepare distinctive outcomes. This is due to the truth that all devices hold fluctuating presumptions something like their metric definitions and accordingly, conclusions reasonably vary

crosswise over one another. Despite this, we might recognize fascinating likenesses the middle of them as specified over table III. Note that HV may be underpinned via special case of the tool, thereabouts may be excluded from similar investigation in the next table 3.

Table 3: Comparative analysis of tools against metrics calculated

Metric	Concluding observations w.r.t CCCC, SM and JHK
Mvg	CCCC calculates the final value is as selects the extreme &class-wise SM calculates module-wise &reports the outcome as extreme complexity. JHK calculates metrics very close to SM. Since no 2 devices agreed to a common value for MVG, we tested programs with one anonymous well-known quality examination device. It authorizes outcome of SM˝s analysis.
Loc	Out of 3, SM runs most positive value of LOC. CCCC counts all curly brackets {,} &non-blank lines as part of LOC where as in the case of label statements, the SM counts non-blank lines only and does not counts curly brackets. JHK counts same under the label LLOC as SM does. JHK diverges from SM in way it counts the statement.
Com	SM reports this metric for rate form, it need been changed over under altered quality in front of entering under table by bringing two other measurements Lines (counting comments) &percent line with remarks as input parameters. CCCC &JHK straightforwardly returns those bring about outright figures What's more advantageous on counter-check. Around all, these metric remains the greater part stable of all.
Nom	CCCC calculates for NOM may be not similar to its counterparts due to it checks number for classes as against others two, which check amount about capacities &methods spanning through every last one of classes in a project. Since a strategy undoubtedly may be toward a better granularity level over a class, we affirm the outcome of SM or JHK investigation in this instance.

The below table 4 shows the outcomes of evaluation metrics.

Table 4: Characteristics of Program

Programs	Size	Logical Complexity	Documentation	Volume
ProgramX	Highest	least complex	Poor documented	Small
ProgramY	Least Size	Least complex	Few comments	Medium
ProgramZ	middle-sized	More complex	Good documented	High

Metrics portray different project Characteristics objectively. They might a chance to be arranged toward their volume alternately size, association around those modules or many-sided nature from claiming stream control in every system module and a considerable measure additional. These estimations ended up additional serious though a percentage critical personal satisfaction qualities Might be further inferred from the base measurements. Over following sub-section, we endeavor should figure particular case such composite metric will demonstrate relative maintainability that may be a most after quality factor of sought for managers of project.

7. MAINTAINABILITY

The work [22] depicted a MCI (Maintainability Code Index) will be a composite metric, which includes an amount of traditional source code metrics in to a particular amount, which signifies comparative maintainability. It will be estimated with specific equation from HV (Halstead volume), MVG, and LOC. The metric initially estimated as follows:

MCI = 161 - 4.2 * ln(aveV) - 0.33 * (aveMVG) – 16.1 * ln(aveLOC)

Where 'ave' is average of measure per module. To rearrange this measure to lie between 0 and 100, it has been normalized as- MCI*=MAX (0,(171-5.2*ln(aveV)-0.33*(aveMVG)-16.2*ln(aveLOC)))*101/171

It computes a value of index among 0 and 101, which signifies the comparative ease of sustaining source code. The higher value proposes enhanced maintainability. The values of MCI estimated for all programs are shows in table 5.

Table 5: MCI Calculation

Programs	Calculating MCI	Output
ProgramX	Max(0,(161 - 4.2 * LN(318.97)-0.33 * 3.74 - 16.1 * LN(11.4))) *	58.83

	101/172	
ProgramY	Max(0,(171 - 5.2 * LN(519.69)-0.33 * 3.5 - 16.2 * LN(18))) * 100/171	54.14
ProgramZ	Max(0,(172 - 5.1 * LN(727.36)-0.33 * 5.4 - 16.2 * LN(22))) * 101/172	51.37

As per table V, Prog A (Bubble sort) need most noteworthy level from claiming maintainability "around the trio and Prog C (Quick sort) will be practically troublesome on support. Prog B (Selection sort) goes amidst the line. One might notice these perceptions come quite in concordance with the project aspects in table IV. Fast kind carries most astounding unpredictability previously, hotspot code, biggest volume Also Subsequently brings about least maintainability list over table V. Sort of bubble will be easiest with program, less intricate and for any rate volume & scores maximum MI. Exchange-off remains comparative to determination sad for a really. Accordingly the algorithmic qualities and resultant qualities uncover that our projects need aid effectively tried for those said measures.

7.1 Other Derived Parameters

Although the characteristics calculated in sec 7 might not straightly describe quality, but they might be used to derive factors representing potential variations to be conducted in last product. Few quality features that might be defined by these code characteristics are following below:

Exactness: Once LOC will be deliberated, it might demonstrate beneficial to derive the other code characteristics like flaws per KLOC. It evaluates the flaw density & finally the exactness that will be the significant quality metric.

Cost & effort of programming: Another beneficial price metrics derivable from LOC will be price of project per KLOC. In case, assuming 2.00 dollars per LOC, the pure price of coding might be estimated for Prog B as 88 dollars. Likewise built on effort of programmer, MVG (degree of complexity), & consequently evaluation of price might be enhanced. The MVG specifies functional coverage breadth of software.

Fault proneness: The main goal of complexity metric will be to calculate components, which are fault-prone. Built on MVG, remaining defect prediction might be prepared. The more difficult framework will be much challenging it will be to check it completely & more error-prone it is.

to confirm their validity.

Devices either report pictures in diverse forms or measure the characteristics at diverse level of granularity that creates them critical to compare deprived of few normalizations. For instance, CCCC calculates the amount of classes for NOM whereas JHK & SM calculates amount of strategies. SM offers COM in percent form whereas others outcome in stable value.

8. THREATS TO VALIDITY

Quality of service for source code will be a multi face model. Similar any evaluation survey, our results will be partial according to whatever primeval information was utilized to generate them. Bias probable traces incorporate representativeness of source code, selection of programming language, selection of devices & their precision programs for diverse metrics on dissimilar programming languages & other capable devices.

9. CONCLUSION

This manuscript requires estimation of 5 software metrics on a group of 3 well recognized categorization approaches with three computerized examination devices. It will be followed by foundation of maintainability index from factor metrics and a brief purpose of other quality features that might be inferred. Certainly software metrics are reasonable devices accessible to managing for decision making determinations and creating them able to taking proactive exploit in instance of potential software crisis by declaring primary indicators to risk prone problems. However, project executives must formulate their program of tailor made metrics to know company’s unique strategic objectives, user’s custom requirement, priorities, and assumptions to entirely use their massive worth. This survey increases prior experimental literature on software metrics validating the connection among quality attributes & software metrics resultant there on providing the advantages and disadvantages on selecting automated devices that are accessible in massive amount.

REFERENCES

[1] Elshoff, J. L., an Analysis of Some Commercial PL/I Programs. IEEE Transactions on Software Engineering, pp. 113–120, June, 1976. https://doi.org/10.1109/TSE.1976.233538

[2] Improvements Needed in Managing Automated Decision making &v computers Throughout the Federal Government, U.S. General Accounting Office, April 23, 1976.

[3] ISO/IEC 9126-1 Software engineering – Product Quality - Part 1: Quality model", 2001.

[4] IEEE Std 1061-1998 "IEEE standard for a software quality metrics methodology, IEEEpublications.

[5] H. Barkmann, R. Lincke, and W. Löwe. "Quantitative Evaluation of Software Quality Metrics in Open-Source Projects". In Proceedings of The 2009 IEEE International Workshop on Quantitative Evaluation of large-scale Systems and Technologies (QuEST09), Bradford, UK, 26-29th May, 2009. https://doi.org/10.1109/WAINA.2009.190

[6] YasunariTakai, Takashi Kobayashi, Kiyoshi Agusa. "Software Metrics based on Coding Standards Violations", In Proc. the Joint Conference of the 21th International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement (IWSM/MENSURA2011) pp.273-278,Nara, Japan, 3-4 Nov. 2011 https://doi.org/10.1109/IWSM-MENSURA.2011.34

[7] Yuming Zhou, BaowenXu, Hareton Leung. "On the ability of complexity metrics to predict fault-prone classes in object- oriented systems". Journal of Systems and Software, 83(4), 2010:660-674 https://doi.org/10.1016/j.jss.2009.11.704

[8] S Pradeep, Chaudhary K D and V Shrish. "An Investigation of the Relationships between Software Metrics and Defects".

[9] Yuming Zhou, BaowenXu, Hareton Leung. 2010. "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems". Journal of Systems and Software, 83(4), 2010:660-674. https://doi.org/10.1016/j.jss.2009.11.704

[10] Roger S. Pressman, "Software Engineering – A Practitioner‟s Approach", 5th Ed., McGraw Hill International Edition.

[11] Cesar Couto, Christofer Silva, Marco Tulio Valente, Roberto da Silva Bigonha, Nicolas Anquetil. "Uncovering Causal Relationships between Software Metrics and Bugs". CSMR 2012:223-232 https://doi.org/10.1109/CSMR.2012.31

[12] Glasberg, D., Emam, K. E., Melo, W., and Madhavji, N., "Validating Object-Oriented Design Metrics on a Commercial Java Application," National Research Council 44146, September 2000.

[13] AK Pandey. "Predicting Fault-prone Software Module Using Data Mining Technique and Fuzzy Logic". Special Issue of IJCCT Vol. 2,2010

[14] Aaron Don M. Africa," A Comprehensive Study on Application Development Software Systems", International Journal of Emerging Trends in Engineering Research, Volume 7, No.8 August 2019. https://doi.org/10.30534/ijeter/2019/03782019

[15] Nagappan, N., Ball, T., Zeller, A. 2006.: Mining metrics to predict component failures. In ICSE(2006)452-461. https://doi.org/10.1145/1134285.1134349

[16] YuhanisYusof and Qusai Hussein Ramadan, 2010. Automation of Software Artifacts Classification. https://doi.org/10.3923/ijscomp.2010.109.115