International Journal of Scientific Research and Engineering Development – Volume 2 Issue 4, July – Aug 2019

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Measurement of Software Metrics

Sanjeev Kumar

----------**********----------

Abstract:

By its nature, engineering is a quantitative discipline. Engineers use numbers to help them design and assess the product to be built. Until recently, software engineers had little quantitative guidance in their work - but that's changing. Technical metrics help software engineers gain insight into the design and construction of the products they build. Software project effort estimation is frequently seen as complex and expensive for individual software engineers. The Metric-set selection has a vital role in software cost estimation studies; its importance has been ignored especially in neural network.

Keywords : Software Robustness, Software Metrics, Software Quality, Project Metrics, Software Economics.

----------**********----------

1. INTRODUCTION

Software engineering describes the collection of techniques that apply an engineering approach to the construction and support products. Software engineering activities include managing , costing , planning modeling , analyzing , specifying designing ,implementing , testing and maintaining. A software metric is an attribute of the software development environment derived from measuring attributes of software objects. It is a quantative measure of the degree to which a system ,component ,or process processes a given attribute .

A software item might be:-

1. A software product or partial product .
2. A software process such as coding or specifying ; an event such as a product failure.
3. A person involved in software production or use such as a designer or a project manager.
4. An organization such as a data processing department or software house.

In 1993 the IEEE published a standard for software quality metrics methodology that has since defined and led development in the field. Here we begin by summarizing this standard. It was intended as a more systematic approach for establishing quality requirements and identifying, implementing, analyzing, and validating software quality metrics for software system development. It spans the development cycle in five steps as shown in table 1.1.

Table 1.1 : IEEE Software Quality Metrics Methodology

Software Quality Activity	Development Cycle Phasing
Establish software quality requirements	—
Identify software quality metrics	—
Implement software quality metrics	—
Analyze results of these metrics	—
Validate the metrics	—

The next Table 1.2 gives the suggested metrics paradigm for a description of modern projects.

Table 1.2 : Recommended Metrics Set for a Modern Project

Category	Management Insight	Management Indicators
Progress	Provides information on how well the project is performing with respect to its schedule.	Actual vs. planned task completions Actual vs. planned durations
Effort	Provides visibility contributions nt he of o on project costs, sta sch adherence, and product ffi edu quality. ng le	Actual vs. planned staffing profiles
Cost	Provides tracking of actual costs against estimated costs and predicts future costs.	Actual vs. planned costs Cost and schedule variances
Review Results	Provides status of action items from life-cycle review.	Status of action items
Trouble Reports	Provides insight into product and process quality and the effectiveness of the testing.	Status of trouble reports Number of trouble reports opened, closed, etc. during reporting period
Requirements Stability	Provides visibility into the magnitude and impact of requirements changes.	Number of requirements changes/clarifications Distribution of requirements over releases
Size Stability	Provides insight into the completeness and stability of the requirements and into the ability of the staff to complete the project within the current budget and schedule.	Size growth Distribution of size over releases
Computer Resource Utilization	Provides information on how well the project is meeting its computer resource utilization goals/requirements.	Actual vs. planned profiles of computer resource utilization
Training	Provides information on the training program and staff skills.	Actual vs. planned number of personnel attending classes

Four other parameters can be introduced to measure quality after the software has been delivered:

- Fix the backlog and the backlog management index (BMI)
- Fix the response time and responsiveness
- Percentage of delinquent fixes
- Fix quality

As dynamic metrics have the advantage of being more precise, but they are difficult to compute in comparison to static ones. Thus, there is clear opportunity for researchers in hybrid approach where the dynamic analysis results can be augmented by static information for collection of metrics data as well as the concept of core metrics must be applied. The comparison of different software techniques is shown in the table 1.3

Table 1.3 : Comparison between Different Software Techniques

Attribute	Conventional Techniques	Intermediate Techniques	Modern Techniques
Size	100% Custom.	30% Component based, 70% Custom	75% Component based, 25% Custom
Tools	Custom	Off the self, Separate	Off the self, Integrated
Process	Ad-hoc	Repeatable	Managed

1. SOFTWARE SIZE METRICS

➢ Lines of Code (LOC):-

The basis of measure LOC is that program characteristics such as effort and case of maintenance.The LOC Is used to measure size of the software. the versions of LOC is DSI(delivered source instruction) and KDSI (thousands of delivered source instruction).the advantage of the use of LOC is that is easy to measure . the drawbacks of LOC are that it is defined on code for example it can not measure the size of specification,functionality or complexity of the code and it is a language dependtdent.

➢ FUNCTION POINTS(FP):-

FP data is used in two ways as an estimation variable that is used to "size" each element of the software and as baseline metrics collected from past projects and used in conjuction with estimation variables to develop cost an effort projections. Unadjusted function point (UFP) is the sum of all occurrences ,computed by multiplying each function count with a weighting and then adding up all the values . the weights are based on the complexity of the feature being counted .albrechi 's original method classified the weightings as shown below.

Function type	Low	Average	High
External Input	3	4	6
External Output	4	5	7
Logical Internal File	7	10	15
External Internal File	5	7	10
External Inquiring	3	4	6

In order to find adjusted FP, unadjusted FP is multiplied by technical complexity factor (TCF) Which can be calculated by

TCF = (0.65*(Sum of factors))/100 .

There are 14 TCF are as follows data communication , performance , heavily used configuration, transaction rate, Online data entry, end user efficiency, online update ,complex processing ,reusability , installation case, operation Case, multiple sites ,facillate change ,distributed fuction.

FP = UFP*TCF

PRODUCTIVITY = FP/person month .

QUALITY = number of defects / FP.

DOCUMENTATION =Number of pages of documentation /FP .

2. PUTNAM AND BOEHM MODEL

Indeed, the early resource prediction models such as those of Putnam and Boehm used LOC or related metrics like delivered source instructions as the key size variable in regression-based models of the form

E = F(LOC)

In the late 1960's LOC was also used as the basis for measuring program quality which normally measured indirectly as defects per KLOC. In 1971 Akiyama published what we believe was the first attempt to use metrics for software quality prediction when he proposed a regression-based model for module defect density in terms of the module size measured in KLOC.

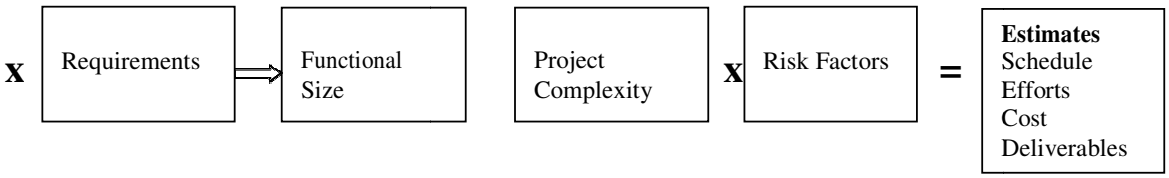

Figure 1.1 : Driving Factors for Software Metrics

3. FUNCTION POINT METRICS

Quality metrics based either directly or indirectly on counting lines of code in a program or its modules are unsatisfactory. Estimating using Functional Metrics :

- Provide a top down estimating approach
- Based on the estimate of customer requirements & constraints
- Consider the technical environment , project characteristics and past performance
- Communicate accurate estimates which can be revised if requirements changes through

A typical function point metric developed by Albrecht at IBM is a weighted sum of five components that characterize an application:

4 x the number of external inputs or transaction types 5 x the
number of external types or reports
10 x the number of logical internal files 7 x the
number of external interface files
4 x the number of external (online) inquiries supported

4. CAPABILITY MATURITY MODEL INTEGRATION (CMMI)

CMMI is the successor to CMM (Capability Maturity Model). Both CMM and CMMI were developed at the Software Engineering Institute (SEI) at Carnegie Mellon University in Pittsburgh, Pa. CMM was developed in the late 1980s, and retired a decade later when CMMI was developed.

There are five levels defined along the continuum of the CMM and, according to the SEI: "Predictability, effectiveness, and control of an organization's software processes are believed to improve as the organization moves up these five levels.

- **Initial** (chaotic, ad hoc, individual heroics) - the starting point for use of a new or undocumented repeat process.
- **Repeatable** - the process is at least documented sufficiently such that repeating the same steps may be attempted.
- **Defined** - the process is defined/confirmed as a standard business process, and decomposed to levels 0, 1 and 2 (the latter being Work Instructions).
- **Managed** - the process is quantitatively managed in accordance with agreed-upon metrics.
- **Optimizing** - process management includes deliberate process optimization/improvement.

5. SIX SIGMA METRICS

Six Sigma employs a measurement driven approach to continuous improvement. It starts with quantitative business goals providing direct value to the customer. Data analysis is used to identify specific processes with the greatest leverage on these goals. Critical inputs affecting process performance are identified. Goals are related to changes in process outputs. If measurements indicate goals have been achieved, improvements are institutionalized. Controlling critical inputs controls process performance to the new levels.

The acronym for the Six Sigma continuous improvement methodology is DMAIC (Define Measure Analyze Improve Control). The DMAIC method is used in conjunction with the Six Sigma toolkit, a more or less standard set of statistical analysis techniques.

Unlike the model based approaches to process improvement such as the CMMI, Six Sigma is not domain specific. Six Sigma evolved out of the manufacturing world, a world with complex proprietary processes. Six Sigma does not include any process models. It is a strategy to improve the bottom-line coupled with a measurement driven method for continuous improvement. This emphasis on method and first principles analysis can be a both strength and a weakness when Six Sigma is applied to software development.

5. ALBRECHT AND IFPUG

(International Function Point Users Group) use a weight table to assign size values to individual function types. No justification is given for the specific selection of those weights, or the selection of functional size indicators. Symons derives the weights for his function types from the effort required to develop the function type. Such correlation between functional size and effort is beneficial for the estimating of development time. However, it limits the validity of the metric when normalizing software, or establishing development productivity.

A formal methodology has been proposed by Wittig to determine empirically the weight coefficients for the IFPUG Function Point method. The IFPUG Function Point method defines Function Point analysis as “a standard method for measuring software development from the customer‘s point of view IFPUG. Wittig et al. use the Analytic Hierarchy Process (AHP) developed by Saaty to establish the weight factors. AHP is a proven technique of multiple criteria decision making. It has been successfully applied in industry, government and research establishments. Comparing five methods for determining weights in additive utility models Schoemaker and Waid found AHP to be the best technique.

The first finding of the study is that software development experience influences function point estimates. This can be understood by comparing the results of the technical and non-technical groups. Although the statistics were based on the analysis of a small number of examples and may not constitute any strong recommendation for management to use FP technique, the experiment can be regarded as a very useful reference to correct the general misunderstanding of using this model.

According to the survey, 34% of respondents currently use Agile, and 12% have plans to implement Agile within the next 12 months. This is down from 57% and 42%, respectively, in 2010. “Agile has been in the mainstream for a while. People understand that it exists. If anything, the rate of adoption has increased,” says Michael Azoff, principal analyst, Ovum. “From our perspective, there’s an increase in adoption and a desire to adopt Agile, whether on an individual project basis or within a whole department”.

Matthew Hotle, vice president and distinguished analyst, Gartner Research, agrees that “The breadth of penetration into our client base is fairly high, but the depth is fairly low,” he says. According to Hotle, upwards of 50% of Gartner’s client base is using Agile, but less than 10% of work being done is actually being done in an Agile way.

6. SOFTWARE MATURATION MODELS

In 1985, Redwine and Riddle presented a maturation model of software technology, which discusses the steps for transition from a research concept to practice. They identified six major phases.

Basic Research : investigate ideas and concepts that eventually evolve into fundamentals and identify the critical problems and sub-problems.

Concept Formulation : key idea or problem is identified and the ideas are informally circulated, convergence of a set of ideas, and publications start to appear that suggest solutions to sub-problems.

Development and Extension : present a clear definition of a solution in either a seminal paper or a

demonstration; preliminary use of the technology; generalize the approach.

Internal Enhancement and Exploration : extend approach to another domain, use technology to solve real problems; stabilize and port technology; development of training materials.

External Enhancement and Exploration : similar to previous phase but performed by a broader group, including outside the development group.

Popularization : show substantial evidence of value and applicability; develop production-quality version, commercialize and market.

7. CYCLOMATIC COMPLEXITY METRIC

Halstead founded software science, and proposed a topological or graph-theory measure of cyclomatic complexity as a measure of the number of linearly independent paths that make up a computer program. To compute the cyclomatic complexity of a program that has been graphed or flow-charted, the formula used is

$M = V(G) = e - n + 2p$

where, V (G) = the cyclomatic number of the graph G of the program e = the number of edges in the graph
n = the number of nodes in the graph

p = the number of unconnected parts in the graph

More simply, it turns out that M is equal to the number of binary decisions in the program plus 1. An n-way case statement would be counted as n.1 binary decisions.

Cyclomatic complexity (McCabe) is used to evaluate the complexity of an algorithm in a method. It is a count of the number of test cases that are needed to test the method comprehensively. The formula for calculating the cyclomatic complexity is the number of edges minus the number of nodes plus 2. For a sequence where there is only one path, no choices or option, only one test case is needed. An IF loops however, has two choices, if the condition is true, one path is tested; if the condition is false, an alternative path is tested.

A method with a low cyclomatic complexity is generally better. Cyclomatic complexity cannot be used to measure the complexity of a class because of inheritance, but the cyclomatic complexity of individual methods can be combined with other measures to evaluate the complexity of the class. Although this metric is specifically applicable to the evaluation of Complexity, it is also related to all of the other attributes.

8. OBJECT ORIENTED METRICS

Increasingly, object-oriented measurements are being used to evaluate and predict the quality of software. A growing body of empirical results supports the theoretical validity of these metrics. The validation of these metrics requires convincingly demonstrating that the metric measures what it purports to measure (for example,
a coupling metric really measures coupling) and the metric is associated with an important external metric, such as reliability, maintainability and fault-proneness. Often these metrics have been used as an early indicator of these externally visible attributes, because the externally visible attributes could not be measures until too late in the software development process.

Some limitations of Object-Oriented Metrics are as follows: ISO/IEC international standard (14598) on software product quality states, "Internal metrics are of little value unless there is evidence that they are related to external quality." It need be noted that the validity of these metrics can sometimes be criticized. Many things, including fatigue and mental and physical stress, can impact the performance of programmers with resultant impact on external metrics. The only thing that can be reasonably stated is that the empirical relationship between software product metrics is not very likely to be strong because there are other effects that are not accounted for, but as has been demonstrated in a number of studies, they can still be useful in practice.

Table 1.4 : Metrics for Object Oriented Systems

Source	Metrics	Object-Oriented Area
Traditional	Cyclomatic complexity (CC)	Method
Traditional	Lines of Code (LOC)	Method
Traditional	Comment percentage (CP)	Method
NEW Object-Oriented	Weighted methods per class (WMC)	Class/Method
NEW Object-Oriented	Response for a class (RFC)	Class/Message
NEW Object-Oriented	Lack of cohesion of methods (LCOM)	Class/Cohesion
NEW Object-Oriented	Coupling between objects (CBO)	Coupling
NEW Object-Oriented	Depth of inheritance tree (DIT)	Inheritance
NEW Object-Oriented	Number of children (NOC)	Inheritance

A class is a template from which objects can be created. This set of objects shares a common structure and a common behavior manifested by the set of methods. A method is an operation upon an object and is defined in the class declaration. A message is a request that an object makes of another object to perform an operation. The operation executed as a result of receiving a message is called a method. Cohesion is the degree to which methods within a class are related to one another and work together to provide well-bounded behavior. Effective object oriented designs maximize cohesion because cohesion promotes encapsulation. Coupling is a measure of the strength of association established by a connection from one entity to another. Classes / objects are coupled when a message is passed between objects; when methods declared in one class use methods or attributes of another class. Inheritance is the hierarchical relationship among classes that enables programmers to reuse previously defined objects including variables and operators.

9. CONCLUSIONS WITH FUTURE SCOPE

The Scope Manager focuses on the effective management and control of the project and uses their metrics skills to provide objective evidence of their observations, shifting the focus from measurement to project governance. In doing so we make observations about the status of the project and the quality of both the requirements and the specifications and based on our experience with similar projects, we are able to make predictions on the likelihood of the project's success.

A metrics program that is based on the goals of an organization will help communicate, measure progress towards, and eventually attain those goals. People will work to accomplish what they believe to be important. Well-designed metrics with documented objectives can help an organization obtain the information it needs to continue to improve its software products, processes, and services while maintaining a focus on what is important. A practical, systematic, start-to-finish method of selecting, designing, and implementing software metrics is a valuable aid.
During working in this area of research, a lot of scope for future has been found and realized. There is need of further empirical investigations of the proposed package level metrics in order to establish their relations with internal and external software quality and quantity factors such as maintainability / reparability and reuse.

10. REFERENCES

[1] O. Seng, M. Bauer M, Biehl, and G. Pache, "Search-based Improvement of Subsystem Decompositions", Proc. Genetic and Evolutionary Computation Conference (GECCO'05), Washington, DC, USA, pp. 1045 1051, 25-29 June 2005.

[2] Kannan Mohan and Balasubramaniam Ramesh. Change Management Patterns in Software Product Lines, Communications of the ACM, v. 49, n. 12, 2006.

[3] S. Misra, "A Complexity Measure based on Cognitive Weights", Proc. International Journal of Theoretical and Applied Computer Science, vol.1, no.1, pp. 1-10, 2006.

[4] Software Engineering Institute, "Framework for Product Line Practice," http://www.sei.cmu.edu/productlines, 2006.
[5] K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: Empirical Study of Object-Oriented Metrics. Journal of Object Technology 5(8), 2006.
[6] Shakti Kumar, K. K. Aggarwal, Jagatpreet Singh: A Matlab Implementation of Swarm Intelligence based Methodology for Identification of Optimized Fuzzy Models. Swarm Intelligent Systems 2006: 175-184.
[7] Volker Gruhn, Ralf Laue, "Adopting The Cognitive Complexity Measure For Business Process Models", 5th IEEE Int. Conf. On Cognitive Informatics (ICCI'06), 2006.
[8] Shakti Kumar, K. K. Aggarwal, Jagatpreet Singh: Particle Swarm for Fuzzy Models Identification. Swarm Intelligent Systems 2006: 149-173.
[9] Kushwaha D.S. and Misra A.K "A Modified Cognitive Information Complexity Measure of Software", ACM SIGSOFT, Vol. 31, No. 1, January 2006.
[10] Jitender Kumar Chhabra, K. K. Aggarwal: Measurement of Intra-Class & Inter-Class Weakness for Object-Oriented Software. ITNG 2006: 155-160.
[11] Hongyu Zhang, Xiuzhen Zhang, Ming Gu, "Predicting Defective Software Components From Code Complexity Measures", 13th IEEE International Symposium On Pacific Rim Dependable Computing, 2007.
[12] S.Misra,"An Object Oriented Complexity metric based on cognitive weights" 6th IEEE international conference on cognitive informatics ICCI 2007
[13] S. Misra and A.K. Misra, "Evaluation and Comparison of Cognitive Complexity Measure", ACM SIGSOFT Software Engineering Notes, vol. 32, no. 2, pp. 1-5, 2007.
[14] S. Misra, "Validating Modified Cognitive Complexity Measure", ACM SIGSOFT Software Engineering Notes, vol. 32, no. 3, pp. 1-5, 2007.
[15] Sylvie Trudel, Measurement for improving accuracy of estimates: the case study of a small software organization, Software Measurement European Forum 2007 (SMEF 2007), Rome, Italy, May 9-11, 2007
[16] Simons, Charles (2007), Advancing Functional Size Measurement – which size should we measure?,

Software Measurement European Forum (SMEF 2007), Roma, Italy, May 9-11, 2007 Dondo, A Fuzzy Risk Calculations Approach for a Network Vulnerability Ranking System, Technical

[17] Memorandum 2007-090, Defence R&D Canada – Ottawa, May 2007
[18] Lawrence Carin, George Cybenko, Jeff Hughes, Cybersecurity Strategies: The QuERIES Methodology,IEEE Computer, Vol. 41, No. 8, Aug. 2008
[19] James Figueroa, Discovery Systems Check Their Own Facts, In the News, IEEE Intelligent Systems, Vol. 24, No. 3, May/June 2009.
[20] D. Mishra and A. Mishra, "Object-Oriented Inheritance Metrics: Cognitive Complexity Perspective", Lecture Notes in Computer Science (LNCS), Springer Berlin / Heidelberg, vol. 5589, 2009.
[21] "Package Coupling Measurement in Object-Oriented Software", Journal of Computer Science and Technology (Springer), vol. 24(2), pp. 273-283, March 2009.
[22] G. Woo, H.S. Chae, J.F. Cui, and J.H. Ji, "Revising Cohesion Measures by Considering the Impact of Write Interactions between Class Members", Information and Software Technology, vol. 51, no. 2, pp. 405-417, 2009.
[23] Binstock, Andrew. "Integration Watch: Using metrics effectively". SD Times. BZMedia. http://www.sdtimes.com/link /34157. Oct. 2010.

