International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 3, March 2015, PP 69-72
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Performance Evaluation of Procedural Metrics and Object Oriented Metrics

P.Ashok Reddy[1], Dr.K.Rajasekhara Rao[2], Dr.M.Babu Reddy[3]

[1]Sr.Asst.Professor of Computer Applications LBRCE-Mylavaram, Krishna Dt., A.P, India
[2]Director, Sri Prakash Educational Institutions, Tuni,W.G Dt.,AP, India
[3]Asst Professor of Computer Science Krishna University, Machilipatnam, A.P, India.

Abstract: *Software metrics are widely accepted tools to control and assure software quality. A large number of software metrics with a variety of content can be found in the literature. Software metrics are widely accepted tools to control and assure software quality. A large number of software metrics with a variety of content can be found in the literature. In this paper, different software complexity metrics are applied to study which software complexity measures are the most useful ones in algorithm comparison, and to analyze when the software complexity comparisons are appropriate. Unfortunately, for meaningful results, all the algorithms have to be developed in the same fashion which makes the comparison of independent implementations difficult.*

Object-oriented (OO) metrics are measurements on OO applications used to determine the success or failure of a process or person, and to quantify improvements throughout the software process. These metrics can be used to reinforce good OO programming techniques, which leads to more reliable code. The process provides a practical, systematic, start-to-finish method of selecting, designing and implementing software metrics. These metrics were evaluated using object oriented metrics tools for the purpose of analyzing quality of the product, encapsulation, inheritance, message passing, polymorphism, reusability and complexity measurement. It defines a ranking of the classes that are most vital note down and maintainability.

Object oriented software development requires a different approach from traditional development methods including the metrics used to evaluate the software. It means that traditional metrics for procedural approaches are not adequate for evaluating object oriented software primarily because they are not designed to measure basic elements like classes objects polymorphism and message passing Even when adjusted to syntactically analyze object oriented software they can only capture a small part of such software and therefore can just provide a weak quality indication.

Keywords: *Metrics, Procedural Metrics, OO Metrics, Software Metrics, Performance Evaluation, Object Oriented Programming Concept, Procedural Concepts.*

1. INTRODUCTION

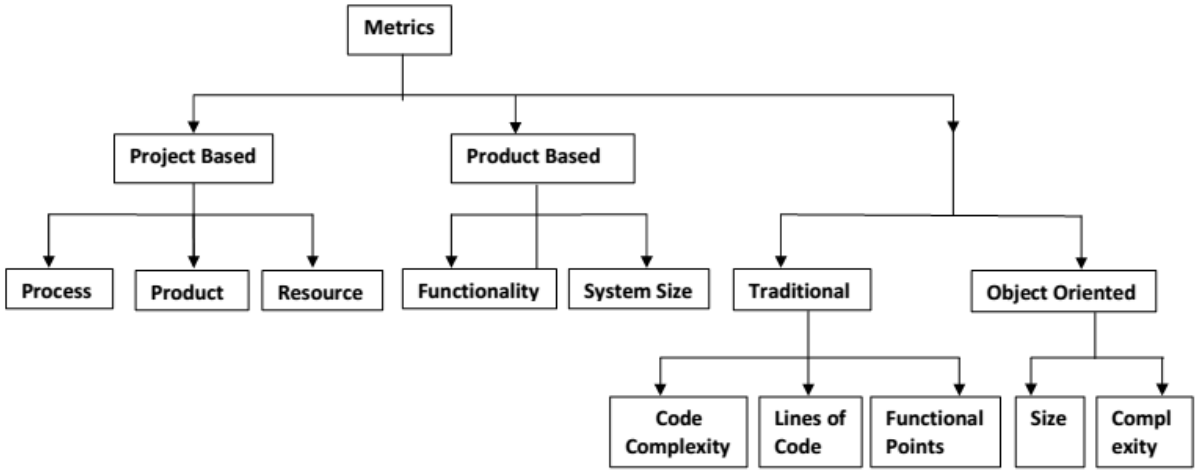
Software Metrics are standards to determine the size of an attribute of a software product and a way to evaluate it. Modern software engineering dictates that software can be organized into a set of modules. A module captures a set of design decisions which are hidden from other modules and the interaction among the modules should primarily be through module interfaces.

2. OBJECTIVE

- To analyze of various procedural as well as object oriented software metrics.
- To select the most useful software metrics.
- To design an automation system that will present the software measurement analysis

3. METRICS

Metrics are used to evaluate the software project. Project based metrics keep track of project maintenance, budgeting etc. Design based metrics describe the complexity, size and robustness of object oriented and keep track of design performance.



4. TYPES OF METRICS

The first rule of quantitative software evaluation is that if we collect or compute numbers we must have a specific intent related to understanding, controlling or improving software and its production. This implies that there are two broad kinds of metrics: product metrics that measure properties of the software products; and process metrics that measure properties of the process used to obtain these products. Product metrics include two categories.External product metrics cover properties visible to the users of a product; internal product metrics cover properties visible only to the development team.

4.1. External Product Metrics Include

- Product non-reliability metrics, assessing the number of remaining defects.
- Functionality metrics, assessing how much useful functionality the product provides.
- Performance metrics, assessing a product's use of available resources: computation speed, space occupancy.
- Usability metrics, assessing a product's ease of learning and ease of use.
- Cost metrics, assessing the cost of purchasing and using a product.

4.2. Internal Product Metrics Include

Size metrics, providing measures of how big a product is internally. Complexity metrics (closely related to size), assessing how complex a product is. Style metrics, assessing adherence to writing guidelines for product components (programs and documents).

4.3. Process Metrics Include

Cost metrics, measuring the cost of a project, or of some project activities (for example original development, maintenance, documentation). Effort metrics (a subcategory of cost metrics), estimating the human part of the cost and typically measured in¬person-days or person-months. Advancement metrics, estimating the degree of completion of a product under construction. Process non-reliability metrics, assessing the number of defects uncovered so far. Reuse metrics, assessing how much of a development benefited from earlier developments.

4.4. Internal and External Metrics

The second rule is that internal and product metrics should be designed to mirror relevant external metrics as closely as possible. Clearly, the only metrics of interest in the long run are external metrics, which assess the result of our work as perceptible by our market. Internal metrics and product metrics help us improve this product and the process of producing it. They should always be designed so as to be eventually relevant to external metrics. Object technology is particularly useful here because of its seamlessness properties, which reduces the gap between problem structure and program structure (the "Direct Mapping" property). In particular, one may argue that in an object-oriented context the notion of function point, a widely accepted measure of functionality, can be replaced by a much more objective measure: the number of exported features (operations) of relevant classes, which requires no human decision and can be measured trivially by a simple parsing tool .

4.5. Designing Metrics

The third rule is that any metric applied to a product or project should be justified by a clear theory of what property the metric is intended to help estimate. The set of things we can measure is infinite, and most of them are not interesting. but this is unlikely to yield anything of interest to product developers, product users, or project managers it was connected to a very precise hypothesis that the simplicity of such interfaces is a key component of the ease of use and learning (and hence the potential success) of a reusable component library.

4.6. Calibrating Metrics

More precisely, the fourth rule is that most measurements are only meaningful after calibration and comparison to earlier results. This is particularly true of cost and reliability metrics. A sophisticated cost model such as COCOMO will become more and more useful as you apply it to successive projects and use the results to calibrate the model's parameters to your own context. As you move on to new projects, you can use the model with more and more confidence based on comparisons with other projects.

5. PROCEDURAL METRICS

Procedure oriented metrics measure different attributes of a project or smaller pieces of code. For example, a metric may measure the number of code lines, the complexity of code or the amount of comments. Traditional/ Procedural metrics have been applied for the measurement of software complexity and size of structured systems.

6. OBJECT ORIENTED METRICS

Procedural metrics do not capture unique aspects of Object Oriented Programs. Object Oriented Metrics plays a pivotal role in the development of fault free software product. Object oriented design and development are popular concept in today's software development. Object oriented design and development focuses on objects as the prime agents involved in the computation; each class of data and related operations are collected into a single system entity. The main advantage of object oriented design is its modularity and reusability. Object oriented metrics are used to measure properties of object oriented designs.

7. PROPOSED VIEW FOR OBJECT ORIENTED METRICS

In this proposed view for object oriented metrics, user may focus on the following parameters

- System and its Implementation
- Set of classes and their Implementation
- Cohesion and Coupling among modules in the classes
- Supporting of Inheritance among the classes
- Reducing number of lines of code
- Maintaining class diagram with appropriate relationships
- Avoiding duplicated code
- Eliminating Code Complexity

8. QUALITY METRICS TOOL FOR OBJECT ORIENTED PROGRAMMING

"Metrics measure certain properties of a software system by mapping them to numbers (or to other symbols) according to well-defined, objective measurement rules. Design Metrics are measurements of the static state of the project's design and also used for assessing the size and in some cases the quality and complexity of software. Assessing the Object Oriented Design (OOD) metrics is to predict potentially fault-prone classes and components in advances"

9. SOFTWARE QUALITY FACTORS IN PROCEDURE AND OBJECT ORIENTED APPROACHES

- Functionality

  The degree to which the software satisfies stated needs
- Reliability

  The amount of time that the software is available for use
- Usability

  The degree to which the software is easy to use
- Efficiency

  The degree to which the software makes optimal use of system resources
- Maintainability

  The ease with which repair and enhancement may be made to the software
- Portability

  The ease with which the software can be transposed from one environment to another

10. CONCLUSION AND FUTURE SCOPE

In this paper both procedural and object oriented metrics are considered. Procedural oriented metrics are not applicable to object oriented systems. For, object oriented systems different approaches are used to measure the size and complexity of code. software metrics for object oriented paradigm embody the complex set of characteristics inherent in large software systems. It seems many complexity features to enable the software engineer to monitor the software development process. There is no single measure that captures all the features of an object oriented software product. As per this, a better approach to measuring object oriented software products is to isolate the features of the product that are of concern to us and develop a suite of measures that shows available features. In the high-level design phase, the suite of metrics can be used.we have concept of measures for cohesion and coupling, which are important attributes of design. A number of object oriented metrics have been proposed in the literature for measuring the design attributes such as inheritance, polymorphism, message passing, complexity, Hiding Factor, coupling, cohesion, reusability etc,.

REFERENCES

[1] A Measurement Model for Detecting the most Suitable Code Component from Object Oriented Repository**.** By Sumit Jain and Mohsin Sheikh in IJCA with Volume 104-Number by Oct 2014.

[2] A.D. Bakar, A. Sultan, H. Zulzalil and J. Din, 2014. Predicting Maintainability of Object-oriented Software Using Metric Threshold. *Information Technology Journal, 13: 1540-1547.*

[3] Reliability Quantification of Object Oriented Software: A Revisit by A.K.Chand and N.Dhanda with ISSN 2321-7782 in IJARCSMS with Volume 2 Issue 11,Nov 2014.

[4] Reliability Measurement of an Object Oriented Design: A Systematic Review by Nidhi Gupta, Dr. Rahul Kumar with ISSN 2277-1581 in IJSET with Vol 3 with Issue 12, pp :1483-1487 during Dec-2014.

[5] Measuring Inheritance Patterns in Object Oriented Systems: the Dynamic Inheritance Ratio Metric by M.Niculescu, Ph.Dugerdil and Blanco.M.Canedo with ISBN: 978-1-4503-3432-7

By IJSE-15 from pages 130-138 during Jan-2015

[6] Software Complexity Metrics: A Survey by Dr.P.Chitt Babu,A.Narasimha Prasad and D.Sudhakar with ISSN : 2277-128X by IJARCSE With Volume 3, Issue 8 and Aug 2013.

[7] *Encyclopedia of Software Development Life Cycle Metrics* (http://www.sdlcmetrics.org/), 10.11.2010

[8] Lanza M., Marinescu R.: *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented System*. Springer, 1st Edition. Edition, 2010

[9] N E Fenton "Software Metrics" Conference Proceedings of on the future of Software engineering ICSE 00(2000) Volume: 8, Issue: 2, Publisher: ACM Press

[10] Roger S. Pressman "Software Engineering-A Practitioner's Approach" 6th Edition, McGraw Hill International Edition pp 466-472

[11] Satwinder Singh, K.S. Kahlon, "Static Analysis to Model & Measure OO Paradigms", SAC, ACM.

[12] S.R Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 20 No. 6

[13] KP Srinavan, Dr. T Devi, "Design and Development of procedure for new object oriented design metrics', IJCA, Vol. 24, No. 8, Jun 2011

[14] Booch, G: "Object-Oriented Analysis and Design with Applications", 2nd ed., Benjamin Cummings, 1994

[15] Priya Walde and S.V.Kulkarni," Metrics for measuring the quality of object oriented software modularization", Emerging Trends in Computer Science and Information Technology - 2012(ETCSIT2012)

[16] S. Chidamber, C. Kemerer, A Metrics Suite for Object Oriented Design, IEEE Trans. Software Eng., 20/6), 2000, pp. 263-265.

[17] A. Abran. Software Metrics and Software Metrology. Wiley-IEEE Computer Society Pr, 2010

[18] Usha Chhillar, Sucheta Bhasin (2011): A Journey of Software Metrics : Traditional to Aspect-Oriented Paradigm, 5th National Conference on Computing For Nation Development, 10th -11th March, 2011, New Delhi, pp. 289-293

[19] Chidamber, S.R., Kemerer, C.F. 1994. A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering. vol. 20.no. 6. 476-493.

[20] Mark Lorenz. 1993. Object-Oriented Software Development: A Practical guide. 1993. Prentice hall, Englewood Cliffs, New Jersey.

[21] Srinivasan, K.P., Devi, T., Thiagarasu, V. 2009. Analysis of Chidamber-Kemerer Metrics for Object-Oriented Design. Proceeding of National Conference on Emerging trends in Computer Science, Avinasilingam University for Women, Coimbatore.

[22] Measure the Reusability of Object Oriented Interfaces in UML Diagrams by ukessays.com.