IJITKM Special Issue (ICFTEM-2014) May 2014 pp. 46-50 (ISSN 0973-4414)

Metrics Used In Component Based Software Engineering

Chander Diwaker[1], Sonam Rani[2], Pradeep Tomar[3]
[1,2]Department of CSE, UIET Kurukshetra University,Kurukshetra, India
[3]Department of CSE , School of ICT, Gautam Buddha University, Greater Noida, India
[1]chander_cd@rediffmail.com,[2]sonamkanwal1000@gmail.com, [3]parry.tomar@gmail.com

Abstract- Component Based Software Engineering is a big concern in Industry. CBSE is a process that follows the principle of design and construction of computer based systems using reusable software components. A component is an independent and replaceable part of a system that performs a clear function in the context of a well defined architecture. It results in better productivity, improved quality, reduction in time spent and cost to develop. Software metrics determine the different aspects of software complexity and therefore play an important role in analyzing and improving the quality of software. These metrics play an important role in guiding the software development and deployment models. Metrics used in component based software engineering are helpful in achieving the quality and managing risk in component based system by checking the factors that affect risk and quality. Metrics are helpful in case of the business systems for retrieving large amount of data. Metrics help the developer in identifying the probable risks so that proper corrective action can be taken. Metrics should be defined in a formal manner because natural language creates problem. Various metrics has been proposed to measure the different attributes of a component like functionality, interactivity, complexity, reusability etc.
Keywords- component based software engineering (CBSE), Software metrics in CBSE.

1. Introduction

CBSE is a process that emphasizes the design and construction of computer based systems using reusable software components. It provides the way of developing very large software systems. It concentrates on both the Commercial-off-the-shelf and in-house components. Component based software engineering has been widely accepted as a new and latest approach to software development. Today's the software systems are very difficult, bulky and unmanageable. This causes in lesser productivity, higher risk management and greater software quality. Software metrics measure different aspects of software complexity and therefore play an important role in analyzing and improving the quality of software [12]. Metrics provide important information on external quality aspects of software such as its maintainability, reusability and reliability [7]. Metrics help in providing the data to the system and increasing the quality of the system. It is greatly helpful in the case of the business systems for retrieving large amount of data. System requires higher quality and do not afford any risks in the system. This could be achieved by the use of the software metrics in the system. These metrics are helpful in achieving the quality and in managing risk in the component based system by checking the factors that affect risk and quality. The metrics play an effective role in guiding the software development and deployment models. The metrics do an important role in highlighting the system. Metrics help developer in identifying the probable risks so that proper corrective action can be taken.

2. A Catalog For Metric Proposal Classification

The taxonomy includes a set of qualitative characteristics plus a quantitative assessment scheme, depending on ordinal scales. The quantitative assessment enforces the desired comparability of proposals. Both, the qualitative and quantitative parts provide a basis for determining the strengths and weakness of each proposal. The taxonomy's characteristics are as Follows:
• **Scope**. This refers to the level of granularity and type of artifacts that are the objectives of the metrics-based assessment proposal. A typical difference is between coarse and fine-grained components. Another point is that while some components are white-box, others are black-box. The scope definition puts the constraints on assessments that can be performed on components.
• **Intent**. A description of the level to which each approach may help in achieving those objectives.
• **Technique**. Technique refers to how the metrics were defined and verified. The metrics explanation method may possibly vary from a purely informal description to a formal definition.
• **Critique**. A qualitative assessment of the most important features of the proposal, including its most motivating aspects, as well as its main limitations also provided.
• **Maturity**. The maturity level of the proposal provides a comparison framework based on the use of ordinal scales to characterize the metrics proposals according to four different viewpoints: the underlying quality replica, the

mapping among metrics and the quality model, the procedure of the metrics description, and the level to which the proposal was validated [2].

3. CBSE Metrics

It can be defined at two levels i.e. system level and component level.

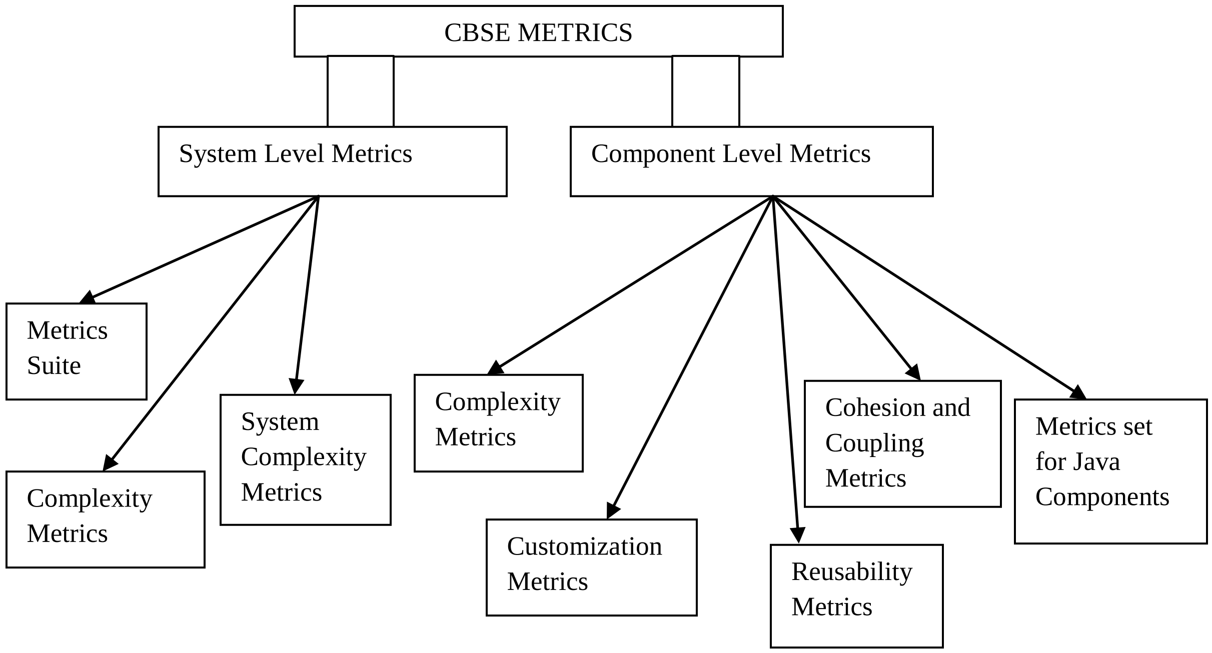

Fig.1 CBSE Metrices

4. System Level Metrics

i. **Metric Suite:-**Metrics for component based systems in three categories: management, requirements, and quality. Management related metrics include cost of product, time to market, maturity of software progress atmosphere, and Software source deployment. Requirement related metrics include: requirement conformance, and requirements permanence. Quality related metrics include: adaptableness, difficulty of interface, incorporation test coverage, and end-to-end test coverage, cumulative number of detected faults, reliability, and customer satisfaction level.

ii. **Complexity Metric:-**In a component based system, complexity results from dependencies among the components of the system. This metric uses inter-component dependency information to determine the complexity of a component based system. Definition of the metric depend on the concept of Component Dependency Graph (CDG) whose vertices represent components, and edges represent the dependencies among components. The CDG is represented in the form of a matrix, with cells of the matrix containing a value 1 if dependency exists in the corresponding component pair and otherwise a value 0

iii. **System Complexity Metric: -** Several metrics are defined to measure a component based system mainly concentrating on its structural complexity. Main attributes that determine complexity of structure of a component based system are identified as: mechanism, connectors, interfaces, and composition tree. The metrics are discussed below:

a) Component Metrics –It includes determining the values of following:
- Total Number of Components (TNC) present in a system.
- Presence of Average Number of Methods per Component (ANMC).
- In a system presence of total Number of Implemented Components (TNIC).

b) Connector Metrics – It includes determining the values of following:
-How many total links are available in a system?

-How many average links are in between components?
- How many average links are available per interface?

c) Interface Metrics – It includes determining the values of following:
- How many average links are available per interface?
- How many average interfaces are existed per component.

d) Composition Tree Metrics – It includes determining the values of following:
- Availability of average number of interfaces per component.
- The width of the composition tree.

5. Component Level Metrics

Component level metrics help in measuring the component quality in terms of its complexity, customizability, and reusability.

i. **Component Complexity Metric:-**It can be Component Plain Complexity (CPC), Component Static Complexity (CSC), Component Dynamic Complexity (CDC), and Component Cyclomatic Complexity (CCC). The CPC metric is the sum of elements of the component (classes, abstract classes, and interface), further complication of all classes, and extra complexity of all methods of the classes. The CSC metric determines the complexity of internal structure of a component. It is the weighted sum of various types of relationships in a component. The CDC metric focuses on the complexity of message passing occurring internally in a component. As compared to other metrics which are available at design stage, the CCC metric is available after implementation. It resembles to the CPC metric, only with a difference that it uses McCabe's complexity metric to determine complexity of method of a class.

ii. **Component Customization Metric:-**It determines the variability of the methods in a component's interface. Metric is the percentage of extent of methods for customization to the total number of methods declared in all the interfaces of the component.

iii. **Component Reusability Metric: -** It measures the reusability of a component at design phase of the component development process. The metric, component reusability (CR) can be calculated as ratio of sum of interface methods providing common functionality in a domain to the total number of interface methods available in a component. The idea to calculate these metrics have been taken from the banking domain. The metrics also provide the indication of some properties of a software component such as understandability, maintainability, and reusability.

iv. **Metrics Set For JAVA Components:-**A set of metrics to measure the reusability property of blackbox components whichare compliant to Java Beans Component model. They define following five metrics for measuringdifferent values that contribute to the reusability of a software component: existence ofmeta-information, component observability, customizability of component, and external dependency.

- EMI (Existence of Meta-Information) – It is a binary metric. It takes a value one if the Bean Info class is provided for a Java Bean component, otherwise it is zero. Availability of Bean Info class enhances the understandability of a component.
- RCO (Rate of Component Observability) – It is determination of percentage of readable properties to the number of fields available in implementation of the class of a component. However if the class does not contain any fields, then value of the metric is zero. High rate of RCO indicates that it is easy to understand the component from the external viewpoint.
- RCC (Rate of component Customizability) – It is calculation of writable properties to the number of fields available in implementation of the class of a component. However if the class does not contain any fields, then metric value is zero. RCO indicates the level of easiness with which a component can be customized for use.
- SCCr (Self-Completeness of Component's Return Value) - It is the percentage of business methods without any return value from all business methods implemented within a component. The metric takes value one when a business method is not present. High value of the metric indicates a low level of external dependency of the component which results in ease of portability.

- SCCp (Self-Completeness of Component's Parameter) - It is the percentage of business methods without any parameters from all business methods implemented within a component. The metric takes value one when a business method is not present. This metric also measures the degree of external dependency of a component. The metrics are combined by the concept of a reusability model which consider that reusability also focuses on understandability, interoperability, adaptability.

v. **Component Cohesion and Coupling Metrics** - The dynamic dependency relationships between classes indicates the high cohesion among components. The cohesion metric takes into consideration the structural relationships along with the types of method called between classes of an object oriented component. Coupling between classes Cm, Cn, denoted by CC (Cm, Cn), is defined as the weighted sum of different types of method called between both the modules of classes. Component coupling is also defined as the sum of coupling among all pairs of classes of the component.

vi. **Contextual Reusability metric** – The metric evaluates a component reusability is different from other metric in this category. The idea is that in addition to internal attributes, component reusability also depends on the context in which it is reused. So their component reusability metric depend on the component's compliance to different elements of the architecture of an application in which it is to be integrated. Metrics belonging to this category are: Architecture observance Metric, and Component observance Metric.

6. Literature Review

Miguel Goulaoet al. [1] presented a metric formalization technique on the basis of use of ontology with a formal specification language. Jianguo Chen et al. [3] suggested a formal direct and indirect component coupling metric for both individual component and assembly between components. P.K.Suriet al. [4] presented the metrics for evaluating the independency of component for reusability. The use of chi-square test has been made for evaluation. V.LakshmiNarasimhanet al. [5] described a systematic comparisons of three suite of metrics allowing a user to choose the best applicable as per the need. P.Edith Linda et al. [6] made comparison among various algorithms on the basis of their performance and memory usage. AbhikritiNarwal[8] defined the complexity metric for software components on the basis of interface methods. SidhuPravneet[9] described an objective way to calculate the quality of software component by using component quality metrics like presence, Ivalues and ratios. These quality metrics have been used to define the exact quality of an artificial intelligence component which is the AI back propagation algorithm. HeshamAbandahet al. [10] presented the effectiveness and power of call graph based metrics by evaluating the many categories of bugs. TaranjeetKauret al. [11] made comparison of various lack of cohesion metrics to increase the fault prediction power and to decrease the complexity. DivyaChaudharyet al. [12] defined various management metrics, requirement metrics, and complexity metrics focusing on various attributes such as cost, quality and productivity.

7.Conclusion

Component Based Software Engineering is the widely used concept in the software industry. Metrics play an important role in determining the various characteristics of a component to find out which components are reusable and what particular function they will perform. Metrics help in providing the data to the system and increasing the quality of system. Metrics are also helpful in managing risk in the component based system. In this paper, a study has been made on how various metrics are used in component based development that concentrates on the factors like complexity, size, reliability, reusability, understandability, maintainability etc. A systematic solution and environment helps the automatic system level measurement. The following are key points that is concluded

- Component characterization is necessary for better understanding of architecture, better usage, better retrieval, better cataloging along with improvement in software reusability.
- Mostly metrics suggested for CBSE has been defined on the basis of theoretical considerations. However the practical paradigm should be considered and theory must be validated.

References

[1] Miguel Goulao, Fernando Brito e Abreu, "Composition Assessment Metrics for CBSE", Proceeding EUROMICRO '05 Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005, pp-96-105.

[2] Miguel Carlos Pacheco AfonsoGoulão, "Component-Based Software Engineering: a Quantitative Approach", Lisboa (2008).

[3] Jianguo Chen, Wai K. YEAP, Stefan D. Bruda"A Review of Component Coupling Metrics for Component based Development" , Proceeding WCSE '09 Proceedings of the 2009 WRI World Congress on Software Engineering - Volume 04, pp. 65-69.

[4] Dr. P.K.Suri, NeerajGarg, "Software Reuse Metrics: Measuring Component Independence and its applicability in Software Reuse" International Journal of Computer Science and Network Security, Vol. 9, No.5, May 2009, pp. 237-248.

[5] V. Lakshmi Narasimhan, P.T. Parthasarathy and M.Das, "Evaluation of a Suite of Metrics for Component Base Software Engineering", Issues in Information Science and Information Technology, 2009, Vol. 6, 2009, pp. 732-740.

[6] P. Edith Linda, V.ManjuBashini, S.Gomathi, "Metrics for Component Based Measurement Tools", International Journal of Scientific and Engineering Research, May 2011, Vol. 2, Issue 5, pp. 1-6..

[7] Gurdev Singh, Dilbag Singh, Vikram Singh, "A Study of Software Metrics", International Journal of Computational Engineering & Management, Jan 2011, Vol. 11, pp. 22-27.

[8] AbhikritiNarwal, "Empirical Evaluation of Metrics for Component Based Software Systems", International Journal of Latest Research in Science and Technology, Dec 2012, Vol 1, Issue 4, pp. 373-378.

[9] SidhuPravneet, "Quality metrics Implementation in Component based Software Engineering using AI Back Propagation Algorithm Software Component", International Journal of Engineering and Management Sciences, 2012, Vol. 3(2), pp. 109-114.

[10] HeshamAbandah and IzzatAlsmadi, "Call Graph based Metrics to Evaluate Software Design Quality", International Journal of Software Engineering and its Applications, Jan 2013 Vol.7, No.1, pp.1-12.

[11] TaranjeetKaur, RupinderKaur, "Comparison of various Lacks of Cohesion Metrics", International Journal of Engineering and Advanced Technology, Feb 2013, Vol. 2, Issue 3, pp. 252-254.

[12] DivyaChaudhary, Prof. Rajender Singh Chillar, "Component Base Software Engineering Systems: Process and Metrics", International Journal of Advanced Research in Computer Science and Software Engineering, July 2013, Vol. 3, Issue 7, pp. 91-95.