UGC APPROVED JOURNAL - 47746
INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 04 , Issue : 03 | January – March 2018

Component-Based Software Metrics: A Review

[1]Parul, [2]Dr. Rajender Singh

[1]Research Scholar, email : parulnarwal8@gmail.com

[2]Professor, email: chhillar02@gmail.com

Department of Computer Science and Application,

Maharshi Dayanand University, Rohtak, Haryana, India

Abstract

Component Based Software Development (CBSD) has been considered a trending technique for software development. This technique focuses on reducing complexity, development time and error factor of software. There are number of software development techniques available but complexity is major issue. Software metrics are capable to play significant role in measuring the complexity of software. Metrics also help in enhancing the quality of software, supporting the developers for finding risks and suggest actions to be performed. In this paper, various component based software metrics reported by various researchers have been reviewed and addressed the issues for future enhancements.

Kew words: CBSD**,** complexity, software components, metrics, system, quality, etc.

1. Introduction

CBSE is a component-oriented approach that stresses the design and development of reusable software components for computer-based systems. Reusability makes CBSE a specialized development technique for the software system. It's a way to create big software systems, of a wide variety. Off-the-shelf, adaptable, and newly developed components are used. Component-based engineering has recently been widely adopted as a new development technique in the field of software. Today's software is complicated, unwieldy, and ineffectual. This decreases productivity, increases risk, and results in lower quality. The complexity of software can be better understood using software metrics. Maintainability, reusability, composability, and reliability can be critical indicators of the quality of software. Metrics support the system by supplying data to the entire system, and by improving the system. It is great for businesses that can retrieve an enormous amount of data quickly. It's necessary to have good software and to make quality projects, but by doing so, you risk going bankrupt. We will use the system's metrics to do this. Metrics are used in improving the output and managing risk in a component-based system. Metrics help developers and administrators identify appropriate software creation and implementation strategies. Metrics

illustrate the framework. The metrics are used to classify potential dangers so that actions can be taken.

2. CBSE Metrics

At one level, it can be described as either system or component

2.1 System-Level Metrics

1. **Metrix Suit :** The metric suite helps to provide the best quality product to the user according to their needs. It manages the quality and cost of the product and also manages the maturity of the software product. It includes Financial and nonfinancial metrics like expense, time to market, the quality of the product, and software engineering development. Quality-related metrics include: Adaptability, testing coverage, the number of errors found problems, and end-to-to-end testing, cumulative testing, and customer satisfaction.
2. **Complexity Metrix:** Complexity is introduced by interdependence between the system's components. This dependency-based metric has uses of inter-component knowledge to look at the system as a whole. The description of the dependency graph depends on the component model, in which the vertices are components and the dependencies bind components together. In CDG projects, each cell represents a degree of dependence. If there is a pair of dependencies, a cell contains a value of 1; otherwise, it contains a value of 0.
3. **System Complexity Metric :** Many metrics are used to quantify structural complexity for a componentizing system. More critical features of autonomous system structures are defined as being those of its nature: how many separate modules are required, how many connectors are needed, and how many interfaces do the system has.

2.2 Component Level Metrics

It tests the component's complexity, as well as customization and reusability component-level metrics.

I Component Complexity Metric:-It can be “Component Plain Complexity (CPC), Component Static Complexity (CSC), Component Dynamic Complexity (CDC), and Component Cyclomatic Complexity (CCC) (CCC). The CPC metric is the number of elements of the variable (groups, abstract classes, and interface), further confusion of all classes, and the additional complexity of all methods of the classes. The CSC metric

calculates the complexity of the internal structure of a variable. It is the weighted sum of different types of relationships in a variable". The CDC metric focuses on the difficulty of message forwarding occurring internally in a component. As opposed to other metrics which are available at the design level, the CCC metric is available after implementation. It resembles the CPC metric, except with the exception that it uses McCabe's complexity metric to calculate the complexity of the system of a class.

II Component Customization Metric:- It tests the customizability of the components that they are customized according to the user's requirement or not. Metric is the percentage of methods to be customized to the total amount of methods declared at all component interfaces.

III Component Reusability Metric: -At the design stage of the part manufacturing process, it checks the reusability of an item. A ratio of sums of interface methods offering standard features in a domain to the total amount of interface methods used in a component may be calculated for metric component reusability (CR). The theory for measuring these measures was derived from the banking domain. The measurements also indicate certain properties of a software product, such as the capacity to learn, maintenance, and reusability**.**

IV Metrics Set For JAVA Components:- The reusability property of black-box components which are true to the Java Beans model is measured in a series of metrics. They define five metrics for assessing various qualities that lead to the re-usability of a software product: Presence of met material, observed component capacity, component customizability, and external dependency.

- The RCO calculates the percentage of readily observed component implementation properties. Without any measuring quantities, the meaning of the metric is negative. A large RCO ensures it is easy to understand from the outside.
- The concept of "meta-information life" is bifurcated. If Bean Info is demanded on a Java bean, then the value is returned, otherwise, 0 is returned. Variable comprehension is improved by the availability of the Bean Info class

- It measures how many fields are writable in an implementation class's resources. When there are no fields in the class, the metric is undefined. RCO reflects the degree of customization necessary.
- SCCr is self-complete on any business operation that does not produce any value. If no market mechanism is involved, the calculation will have no significance. Since it is highly interdependent on external capital, the portability of the commodity is strong.
- The percentage of business methods in the variable still lacking is the auto coverage of the feature parameter. If there is no business formula, the worth is 1. This measure often quantifies the degree of external dependency. When it comes to creativity, the concepts of reusability, adaptability, and capacity are applied to measurements.

5. Component Cohesion and Coupling Metrics : When you hear from using external features, i.e. the components implemented from different staff in-house, take care of coordination and measurements to better think about them. The interconnections of the groups demonstrate the strong connectivity of these components. Cohesion metrics take account, with an object-oriented approach, of hierarchical relationships and methodologies utilized by various object classes. The number of different methods for each category is defined by CC (Cm, Cn). For each class of the variable, the total number of their interconnections is specified.

6. Contextual Reusability Metrix : Contextual reusability differs from other criteria of reusability. In addition to the idea being reusable in the project, the materials are also context-sensitive and this must therefore be taken into account when processing them. The characteristics of application architecture affect the measure of reusability. The following are in this list Architecture indicators, for example, service third parties, as well as component metrics, for example, utilities.

3. Review of literature

(Diwaker, Rani, & Tomar, 2014) studied "*Metrics Used In Component Based Software*" They noticed that Component-Based architecture is the industry standard practise under which most software is designed today. To figure out if an item can be reused, different metrics are needed. metrics support the system by supplying data and building the

consistency of the system The measurements in the component-based method aid in monitoring risk.

(Narang & Goswami, 2018) studied "*Comparative Analysis of Component Based Software Engineering Metrics*" As a process or procedure, it is also known as Component Assembly, Component Linkage, and Component Engineering. When we choose and pick and choose any component from the market, we must be able to quantify their respective qualities, features, and capacity to be interchangeable.

(Gehlot, 2019) studied "*Complexity Metrics for Component Based Software — A Comparative Study*" It also discovered that software metrics play a very significant role in the evaluation and prediction of different software characteristics such as complexity, reusability, maintenance, testability, etc. The complexity of these characteristics impacts all other software attributes. Software Complexity measurements are very important since they are indicative of the potential of future software development improvement. Higher value of complexity increases testing, maintenance and reuse work. One of the most significant concepts is component-based software development (CBAD).

(Aloysius & Maheswaran, 2015) studied "*A Review on Component Based Software Metrics*" and it has noted that CBSD is one of the major contemporary paradigms and is anticipated to be at the forefront of the new approaches to big and complex software systems. This strategy is aimed primarily at minimising development effort, time and costs via reuse and also improving software quality, productivity and maintenance. The reuse of previously integrated software components provides these benefits in particular. A software component is an autonomous software component that delivers explicit functionality, open interfaces and plug-and-play services.

(Tech, 2015) studied "*Quality Enhancement in Reusable Issues in Component – Based Development*" It was discovered that CBD has garnered significant attention from software developers, suppliers and IT companies. A marketplace is developing for software components. Component-based development developed from earlier paradigms of design and programming. CBD is both a subset of existing software engineering techniques and an extension of them. The promise of improved product dependability and stability with shorter development periods and lower costs continue to boost CBD's continuing interest. In order to quickly build and deploy complex software systems with a minimum of engineering work

as well as resource expense, the strategy promotes the purchase, adaption and integration of reusable software products, including COTS products.

(Patel & Kaur, 2016) studied "*A Study of Component Based Software System Metrics*" He noted that nowadays software complexity are growing, making conventional software development tools and techniques inadequate. As a consequence, the ultimate quality decreases. Component Based Software Engineering is the latest method to create software. CBSE is a method that complies with design and construction standards utilising the reusable computer-based system software component. It utilises in-house components and commercial off-the-shelf (COTS) components.

(Tiwari, 2021) studied "*Component-Based Software Engineering*" and that has been discovered in recent years, the nature of software and the development process of software have evolved considerably. Development methods have gone a long way from ad hoc development to a customer-specific agile development process, from basic, one-task programmes to sophisticated, very large-scale software built on components, from simply numerical mathematical computations to real-life solutions. Software is not just a product in today's era: it is a medium for supplying goods, it is a service. In computer science, algorithms and data are two basic structures for creating a programme or software (structure).

(Jena, 2020) studied "*Automated Software Testing Foundations, Applications and Challenges*" It also noted that advances in information and communication technology (ICT) have made it easier for users to communicate more quickly. This method enables to provide citizen-centered services at the doorstep of the population, which benefit distantly situated users greatly. In the past traditional style of government provided its citizens with services via manual mode with a direct participation of human resources, which faced problems of performance throughout its operation. For example, there are human errors during record management, loss, damage or misplacement of official records during the course of operations, delays in service delivery due to physical constraints in message communications, the commitment of huge staff, resulting in repeated costs on state level, chances of favouritism during service delivery, etc.

(Chhillar & Gahlot, 2017) studied "*An Evolution of Software Metrics: A Review*" And it was discovered that software metrics are used to measure different software characteristics such as size, complexity, dependability, quality, etc. They serve a significant role in the analysis

and quality improvement of software. Software metrics also offer valuable information on the software's external quality characteristics, such as maintenance, reusability and dependability. They are also helpful in calculating the testing efforts. Software metrics are broadly classified as product and process metrics. Process metrics are used to measure the process characteristics of the programme.

(Kumar, 2017) studied "*Design of Dynamic Metrics to Measure Component Based Software"* and noted that CBSD integration of dependent and independent software components created and constructed software systems. Assembly of the constituent component software modules plays an important part in the implementation of the programme. This connection enables to assess the software's performance and quality (QoS). The creation of new dynamic measures in CBS enhances performance and quality. These dynamic measurements assist to assess various aspects, such as reusability, simplicity, usability and component integration.

(Voas, 2014) studied "*The Challenges of Using COTS Software in Component-Based Development*" And we discovered that we are progressively drifting towards the promise of broad reuse, where almost any new system can be derived from old code. As a consequence, more and more companies use software not only as all-inclusive programmes as in the past but also as part of bigger applications. The software obtained in this new position must be integrated with existing software functions. This reuse trend has grown so prevalent that it has even altered the language of our business. For instance, the software we purchase elsewhere is characterised by words like commercial-off-the-shelf software (COTS), third-party software (CAS) and no development item (NDI).

4. Conclusion

This paper provides a thorough survey of Component Based software metrics. 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.

Component-based software technology is the idea extensively utilised in the software business. Metrics play a vital role in identifying the different features of a component to determine which components are reusable and the specific purpose. Metrics assist to provide the system with data and improve system quality. Metrics are also useful in the component-based system risk management. In this article we examined how different metrics are utilised

in the creation of components that focusses on aspects such as complexity, size, dependability, reusability, understandability", etc.

References

Aloysius, A., & Maheswaran, K. (2015). A Review on Component Based Software Metrics. *Intern. J. Fuzzy Mathematical Archive Vol. 7, No. 2, 2015, 185-194 ISSN: 2320 –3242 (P), 2320 –3250 (Online) Published on 22 January 2015 Www.Researchmathsci.Org*, *7*(2), 185–194.

Chhillar, R. S., & Gahlot, S. (2017). An evolution of software metrics: A review. *ACM International Conference Proceeding Series, Part F131200*, 139–143. https://doi.org/10.1145/3133264.3133297

Diwaker, C., Rani, S., & Tomar, P. (2014). Metrics Used In Component Based Software Engineering, *50*(May), 46–50.

Gehlot, S. (2019). Complexity Metrics for Component Based Software — A Comparative Study. *Journal of Computers, Volume 14, Number 6, June 2019*, *14*(6), 389–396. https://doi.org/10.17706/jcp.14.6.389-396

Jena, A. K. (2020). *Automated Software Testing Foundations, Applications and Challenges, Services and Business Process Reengineering, ISSN 2524-5503, ISBN 978-981-15-2454-7*.

Kumar, P. (2017). Design of Dynamic Metrics to Measure Component Based Software. *International Conference on Computing, Communication and Automation (ICCCA2017) Design, ISBN: 978-1-5090-6471-7/17/$31.00 ©2017 IEEE*, (May 2017), 753–757. https://doi.org/10.1109/CCAA.2017.8229922

Narang, K., & Goswami, P. (2018). Comparative Analysis of Component Based Software Engineering Metrics. *Proceedings of the 8th International Conference Confluence 2018 on Cloud Computing, Data Science and Engineering, Confluence 2018*, (January 2018), 1–6. https://doi.org/10.1109/CONFLUENCE.2018.8443016

Patel, S., & Kaur, J. (2016). A study of component based software system metrics. *2016 International Conference on Computing, Communication and Automation (ICCCA), ISBN:978-1-5090-1666-2/16/$31.00 ©2016 IEEE*, (April), 824–828. https://doi.org/10.1109/CCAA.2016.7813853

Tech, L. K. M. (2015). Quality Enhancement in Reusable Issues in Component – Based Development. *International Journal Of Research In Electronics And Computer*

Engineering A UNIT OF I2OR (IJRECE) VOL. 3 ISSUE 1 JAN-MAR 2015 ISSN: 2393-9028 | ISSN: 2348-2281, *9028*, 22–26.

Tiwari, U. K. (2021). *Component-Based Software Engineering*.

Voas, J. M. (2014). The Challenges of Using COTS Software in Component-Based Development. *Computer*, *31*(6), 44–45. https://doi.org/10.1109/MC.1998.683006