Direct Research Journal of Engineering and Information Technology
Vol. 8 (4), Pp. 70-73, September 2021
ISSN 2354-4155
DOI: https://doi.org/10.26765/DRJEIT810264375
Article Number: DRJEIT810264375

Original paper

Re-usability Metrics for Black Box Component Based Software Development

Olufunso Oluwasegun

Computer Engineering Department, Delta State Polytechnic, Otefe-Oghara, Delta State, Nigeria.

Author E-mail: sogbaikeoluwasegun@yahoo.com

Received 2 August 2021; Accepted 6 September 2021; Published 30 September 2021

ABSTRACT: Software re-usability is the extent to which a software component can be deployed during component based software development. The choice of component to be selected for reuse requires measuring the software component re-usability. This paper proposes a re-usability metrics that can measure black box component re-usability. Complexity, understandability and custom ability are the properties used for measuring component re-usability. The goal of this paper is to present a re-usability metrics for black box component based software development.

Keywords: Re-usability metrics, black box component, software reuse, software development, component based software development (CBS)

INTRODUCTION

In component based software development, choosing a software component for composition is a foundation for software system development. In component based software engineering (CBSE) activities, issues related to component integration play a more pivotal role than others, and the process of selecting a required component for composition is the challenge of component integrators (Somerville, 2011). A component assembler (software engineer) starts with application requirements, searches component repositories (library) for selecting appropriate components, and assembles them by providing the required attach. From a component assembler perspective, being able to assess the re-usability of candidate component is crucial. A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. Software component can be deployed independently and is subject to composition by third parties; Khimta et al (2008). A generally accepted view of a software component is that it is a software unit with provided services and required services (Lua and Wang 2007).

The main idea of the component-based software development approach is building systems from already existing components. This assumption has several benefits: enhance efficiency, enhance the ability to reuse components, managing growing complexity, reducing the time and effort needed to develop software, decreasing production costs through software reuse, enhancing the quality of system, reducing maintenance costs, increasing development productivity (Siham and Roudies, 2015). Software metrics seeks to represent software characteristics and qualities quantitatively and can be used to direct decision as to whether a component can be selected for composition or not. Several metrics has been deployed to measure and qualify software component re-usability for composition. Hristov et al. (2012) discovered that Measuring re-usability in component based software development requires two categories which includes one for white-box (allowing to look into the code of the components) and the other one for black-box (where usually merely interface and documentation of a component are available) re-usability. The focus of this paper is to outline re-usability metrics

for qualifying black box component for reuse. Black box Components entities encapsulate their services behind well-defined interfaces (Sharma et al., 2007a).
From literature three properties or characteristics that can be used to define black box component re-usability are:

1. Complexity
2. Understandability
3. Customability

Component complexity as defined in Akwukwuma and Sogbaike (2016) is characterized by four parameters which include:

Software Component constituent and interaction: A software component is constituted with several methods, variables, interfaces. Software component complexity metrics should consider these parameters as inputs to the metrics.

Data type of component's value returned or passed and incompatibility: Software component can receive a data type that is passed different from the data type it can handle this result to data-type incompatibility. Data type incompatibility is a factor that causes component complexity.

Coupling: Two objects are coupled if and only if at least one of them acts upon the other. Composing a software system requires some measure of coupling but the degree of coupling translates into the level of dependency of one software component on the other. Coupling is an important parameter when composing software system so as to determine the level dependence of one component on another. This determines the level of complexity of the software component.

Cohesion: Software component that are viewed as architectural units consist of methods. Cohesion of methods in the software component is important when determining a component complexity

$IC = \sum_{i=1}^{m} IMCM_i$ ..... (1)

Interface Method Complexity Metric (IMCM)

IMCM = Wr + PCM (M) + Number of parameter incompatibilities ….. (2),
Where Wr is the weight assigned to return type, PCM (M) is the Parameter Complexity Metric for method.

$PCM\ (M) = \sum_{i=1}^{n} Wp(Pi)$ ..... (3)

Where Wp (Pi) is the weight assigned to the ith parameter of the method on the basis of its data type , n represents the number of parameters in a method in (Table 1). The methods having no return value and no parameters have been considered as simple methods and their weight value has been assumed 0.025.

Sole component complexity metric (SCCM) is combination of the above three component metrics with different weights for each metric.

$$SCCM_j = \alpha * MV_j + \beta * COM_j + \gamma * AIM_j$$
..... (4)

Actual Interactions Metric (AIM)

$AIM_j = \frac{II + OI}{II_{jmax} + OII_{jmax}}$ ..... (5)

II_{jmax} and OI_{jmax} are maximum numbers of input and output interactions in a component j. (OI) and (II) are the Available number of incoming and outgoing interactions
Coupling Metrics

$MV_j = \bigcup_{1 \le i \le m \cap i \ne j} MV_{ji}$ ..... (6)

$$M_j + V_j$$

where M_j and V_j is the sets of methods and instance variables of component C_j. $MV_{j,i}$ is the set of methods and instance variables in component C_i invoked by component C_j. MV_j, the set of all methods and instance variables in other components, $1 \le i \le m$ and $i \ne j$, that are invoked by component C_j

Cohesion Metrics

$COM_j\ (C) = \frac{E_j(C)}{V_j(C) * M_j(C)}$ ..... (7)

Set of method members $M_j(C) \equiv \{ m_{j1}, m_{j2}, \ldots m_{jm}\}$ and a set of instance variables $V_j\ (C) \equiv \{v_{j1}, v_{j2}, \ldots v_{jn}\}$. $E_j\ (C)$ is the set of pairs (v_j, m_j) for each instance variable v in V(C) that is used by method m in $M_j\ (C)$

The hybrid component complexity metrics is defined as follows

HCCM= $\mathbf{IC + MVj + COMj}$ **..... (8)**

Component understandability; understandability can be defined as the capability of the component to enable the user to understand whether it is suitable and how it can be used for particular tasks and conditions of use. Component understandability depends on how much

Table 1: Represents weight values assigned to different categories of data types for parameters and return values.

Parameters/ Return values	Very Simple	Simple	Medium	Complex	Very Complex
Weight assigned	0.10	0.20	0.30	0.40	0.50

Source: Component Complexity Metrics (Chen et al., 2011)

component information is provided for functional description and how well it is documented (Sharma et al., 2007b) component understandability also mean Understanding the functionality of the component, to decide whether it meets the new functional requirements. A user needs high understandability to do this activity. Understandability is defined based on the estimated effort needed by a user to recognize the concept behind a component and its applicability (Washizaki et al., 2003).
Washizaki et al. (2003) defined understandability by two metrics existence of meta information (EMI) and rate of components observability (RCO). Existence of Meta-Information (EMI) checks whether the Bean Info class corresponding to the target component C is provided. The metric can be used by the users to understand the component's usage.
Rate of Component's Observability (RCO) is a percentage of readable properties in all fields implemented within the Façade class of a component C. The metric indicates that high value of readability would help user to understand the behavior of a component from outside the component

$$RCO(c)| = \begin{cases} \frac{P_r(c)}{A(c)} & (A(c) > 0) \\ 0 & (\text{otherwise}) \end{cases}$$

where:
$P_r(c)$: number of readable properties in c
$A(c)$: number of fields in c's Facade class

Customability: Rate of Component's Customizability (RCC) is a percentage of writable properties in all fields implemented within Façade class of a component C. High value of the metric indicates the high level of customizability of component as per the user's requirement and thus leading to high adaptability. But if a component has too much writable property, it will loose the encapsulation and can be used wrongly (Sharma et.al 2007b). Customizability of a component represents the available writable properties within exterior side classes of a component. The components can provide customizable features to enhance reuse spectrum. A component should be customizable during the integration to adjust itself into specific requirements. But more writable properties can be used wrongly. Component's customizability effects on re-usability of components in CBSD (Singh and Tomar 2014). Customability is given as

$$RCC(c) = \begin{cases} \frac{P_w(c)}{A(c)} & (A(c) > 0) \\ 0 & (\text{otherwise}) \end{cases}$$

where:
$P_w(c)$: number of writable properties in c

The proposed re-usability metrics for black box component based software development(BBCBSD) is expressed as

BBCBSD= HCMM + EMI + RCO + RCC

By the above expression, qualifying a black box component for reuse will require using the BBCBSD metrics.

Conclusion

Software re-usability is an essential aspect of component based software development where software systems are developed by integrating existing software components. Qualifying a software component for reuse requires measuring the re-usability of a software component. This paper presents a reliable re-usability metric that considers three properties of re-usability which include component complexity, component understandability and component customability. Using the proposed re-usability metrics, software components can be reliable selected developed with minimal selection challenges.

REFERENCES

Hristov, D., Hummel, O., Huq, M., and Janjic, W. (2012). Structuring Software Re-usability Metrics for Component-Based Software Development, ICSEA: The Seventh International Conference on Software Engineering Advances Pp. 421- 429.
Khimta, S, Sandhu, P. S., and Brar, A. S. (2008). A Complexity Measure for JavaBean based Software Components: World Academy of Science, Engineering and Technology Pp. 449-452.
Lua, K. and Wang Z. (2007). Software Component Models; IEEE Transactions on Software Engineering, (33)10 pp 709 – 724.
Sharma, A., Kumar R., and Grover P. S. (2007a). Managing Component-Based Systems With Reusable Components International Journal of Computer Science and Security, 1 (2): 52-57.
Sharma, A., Kumar, R., and Grover, P. S. (2007b). A Critical Survey of Re-usability Aspects for Component-Based Systems World Academy of Science, Engineering and Technology International Journal of

Social, Behavioral, Educational, Economic, Business and Industrial Engineering (1):9, pp 420 -424.
Siham, Y., Ounsa, R. (2015). All about Software Re-usability: A Systematic Literature Review Journal of Theoretical and Applied Information Technology (76):64-75.
Singh, A. P., and Tomar, P. (2014). World Estimation of Component Re-usability through Re-usability Metrics Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering (8):11.
Somerville I. (2011). Software engineering Addison Wesley publication ninth editions.
Washizaki, H., Hirokazu, Y. and Yoshiaki, F. (2003). A Metrics Suite for Measuring Re-usability of Software Components Proceedings of the Ninth International Software Metrics Symposium IEEE computer society.