International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Analysis and Estimation of Interaction and Reusability Complexities of Components in Component-Based Software Engineering

Nidhi Mehra, Divya Kapil

Abstract: ***Reuse is the elementary and essential attribute of the component-based software engineering. Reusability focuses on building huge and complex software through assembling pre-fabricated software constructs. In this paper, our emphasis is on to analyze and quantify the two core notions of component-based software, that is, reusability and interaction behavior of the components with the goal of minimum complexity generation. We analyse prominent works available in the literature in the area of reuse and interaction complexities. We Analyze and Estimate reusability complexities of components in Component-Based Software Engineering and we propose efficient and useful metrics to compute the reusability of components. Reusability metrics are defined for individual components as well as overall system level. Different types of components are defined on the basis of their reusability, that is, components that can be reused as it as, and tailored components. To define the metrics we have used lines-of-code as the basic parameter due to its simplicity and countability. We proposed a matrics which select the best component on the basis LOC based complexity.We compare IRMCcTotal-i matrics with SCCp metrics and SCCr metrics and analyses that negative relationship between SCCp and IRMCcTotal-i shows low portability and positive relationship between SCCr and IRMCcTotal-i shows high portability.***

Keywords: CBSE, Reusability, Interaction, Complexity metrics.

I. INTRODUCTION

As a process software engineering includes documentation, design, program, test, and maintenance which can be measured statistically. The prime requirement of this process is to efficiently monitor the quality of software .Early prediction of software cost and quality is important for better software planning and controlling. In early development phases, design complexity metrics are considered as useful indicators of a software testing effort and some quality attributes. Due to the appearance of certain factors that affect the quality of software, many practitioners believe that there is a direct relationship between internal attributes such as cost, effort, LOC, speed or memory and external software product attributes such as functionality, quality, complexity, efficiency, reliability or maintainability [1]. For example, a higher number of lines of code or code lines will lead to greater software complexity and so on.

With the ever-increasing complexity of software applications and reducing development time, the software quality and trustworthiness have become an essential concern of the software business. To deal with the software quality, its development time and cost, a Component-Based Software Development (CBSD) is the best approach to be adopted. Component-Based Software Engineering (CBSE) is best practice to reduce the complication of the process. The primarily concerned behind the software development from pre-developed modules is to implement the concept of reusability in appropriate interacted manner. Its major objective is to reduce the software development cost and time by reusing available components, including third-party and internal developed components. CBSD demands high quality components to achieve its goals achieve its goals, which requires effective reusability and interaction metrics.

Reusability

Implementing the concept of reusability means developing software systems (or components) by assembling prefabricated components. In order to identify prefabricated components efficiently, a reusable quality base with results from previous software development activities is needed.

The intention behind the concept of Component-Based Software Development (CBSD) is to increase the basic characteristic of software i.e reusability. CBSD is a software engineering methodology that aims to design and develop software systems using reusable components [2, 3, 4, 5, 6, 7,8]. Components are used as the basic building blocks in CBSD. Rather than developing new code, in this paradigm, we reuse and integrate existing components. Integration of these components is based on defined specifications and design architectures using predefined interfaces [1, 9, 10, 11, 12,13]. Interaction among the components in Component-Based Software Development Integration and Interaction among the components according to the user's specification of requirements are very difficult to implement because software components are usually designed and developed in an isolated environment. When these components deploy later in another software system, software components may need to be configured or customized in a different environment.

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Nidhi Mehra School of Computing Graphic Era Hill University, Dehradun, India. E-mail: nidhigehu@gmail.com,
***Divya Kapil**, School of Computing Graphic Era Hill University, Dehradun, India. E-mail: divya.k.rksh@gmail.com

Therefore, when analyzing the interaction among the components, we can expect to encounter more uncertainties than with traditional software systems.

We classify software metrics on the basis of their inception in software life cycle. The arrangement of 8 popular metrics sets, which are listed in the [14] is shown in Table 1.

Table 1

Metrics	Analysis	Design	Coding	Maintenance
McCabe	Yes	Yes	Yes	No
Halstead	No	No	No	No
Line Of Code	No	No	Yes	No
Error Count	No	No	No	Yes
Object-Oriented Class Metrics	Yes	Yes	Yes	No
Software Package Metrics	Yes	No	No	No
Cohesion	Yes	No	No	No
Coupling	Yes	No	No	No

II. BACKGROUND

A. Reusability

Component-Based Software Engineering addresses the expectations and requirements of the customers and users as other branches of Software Engineering do. It follows all the development steps and phases as other development paradigms. The standard Software Engineering principles are applicable to the applications developed through Component-Based Software Engineering. Reusability presents the development team to concentrate on the quality aspects of the software [15]. Component-Based Software development emphasizes "development with reuse" as well as "development for reuse". Development with reuse focuses on the identification, selection, and composition of reusable components. The property of reusability is not applied only to develop the whole system but also to develop the individual components. The development for reuse is concerned with the development of such components that may be used and then reused in many applications, in similar and heterogeneous contexts. Review work is broadly divided into two issues, namely, reusability of components, interaction and interaction complexities

Reusability is the most important aspect of Component-Based Software Development. Software reusability defines the effective reuse of pre-designed and tested parts of experienced software in new applications. In CBSE, we integrate components of all classes according to the design architecture and applications requirements. There are some components for which code is not available called Black-Box components; some components may be available with their code and documentation, which are known as White-Box components. In literature, various researchers have classified software reusability quantification methods into different categories.

Prieto et al. [14] defined some attributes of a program and related metrics to compute reusability in their work. They proposed that reusability metrics on the basis of size and program structure, along with documentation and programming language reuse experience. Further, they used lines of code to count the size, Cyclomatic complexity for structure, rating from 0 to 10 for documentation, inter-module language dependency to estimate the difficulty of modification and experience of using same module.

Caldiera and Basili [16] proposed one of the earliest methods to identify and qualify reusable components. They defined cost, usability, and quality as the three factors affecting the reusability. They characterized components reusability using four metrics: Volume like operands and operators, using Halstead Software Science Indicators. Cyclomatic complexity using McCabe's method to compute the component's complexity, Regularity measures the component's implementation economy, and Reuse occurrence frequency that is not direct measure of the functional usefulness.

Chen et al. [17] presented their findings based on a large number of reused components. They computed size, program volume, program level, difficulty to develop, and effort for all these reused components. Their conclusion is that to increase the productivity we should decrease the values of these metrics.

Gregory [18] defines the theory of function, form, and similarity to compute the software reusability. Function defines the actions of a component, form characterizes the attributes like structure and size and the similarity identifies the common properties of components. The author uses the well-defined metrics like McCabe's complexity metric in his calculation.

Lee and Chang [19] suggested metrics including the complexity and modularity of components to guess the reusability and maintainability in object-oriented applications. They defined complexity metrics as Internal-External Class Complexity and modularity metrics as Class Cohesion Coupling.

B. Interaction and Integration complexities

Jianguo Chen, Hui Wang, Yongxia Zhou, Stefan D. Bruda [2] investigate and describe enhanced quantifiable tools and techniques for using effective software metrics for modern Component-BasedSoftwares Systems

Slyngstad N. P. O. [11], reported that the results of an investigation on the defect, bugs, errors, fault, failure and change densities for individual components in a model of reusable software components. This study contributes towards our investigation of software evolution impact on individual reusable components.

Table 2 interaction and reusability metrics

S.No	Paper	Parameter	Sub Parameter	Interaction Metric
1	[20]	component quality analysis	testing, debugging and maintenance	Component Interface Complexity Metric (CICM)
		classes or use cases		Component Size Metric (CSM)

		Software Expenses	costs of component acquisition and integration	Component Cost Metrics (CCM.
2	[21]	Interaction between components	Incoming interactions(I_A) and Outgoing Interaction(O_A)	Component interaction metric (CIM)

		Interaction density among components	I:Input interactions, O:Output interactions, $I_{Max:}$maxno.of input interaction $O_{Max:}$maxnooutput interactions	Actual interactions metric (AIM)
				Total interactions performed metric(TIPM)
				Complete interactions metric (CpIM)
		Coupling		Direct component coupling metrics (DCMCM)
		Coupling		Indirect component coupling metrics (IDCMCM)
3	[22]	Interaction	Incoming and Outgoing interaction	% age of Component Interactions (CI %)
		Interaction	Actual no of interaction to particular component	Interaction %age Metrics for Component Integration (I%MCI)
		Interaction	Performed Interaction	Actual interactions (AI)
		Interaction	Total no of components	Total Interactions Performed (TIP)
		Interaction Complexities	Incoming & outgoing interaction complexity	Complete interactions in a CBS (CI)
4	[23]	Components Of System	Graph G=(N,E) N =vertices (called components) and E =Edges the set of dependencies (path) between the components	Component Dependency Metric (CDM)
		Components of system		Dependency oriented complexity metric (DOCM)
	[24]	Coupling	No of request for the service among the components	Component Interaction Density Metric (CIDM)
6	[25]	Reusability		System coupling metrics (SCOUP)
		Reusability		System cohesion metrics (SCOM)
		Reusability		System actual interface metrics (SAIM)
7	[26]	Coupling	FICM(BB)= Fan-in Complexity Metric(measures the coupling complexity due to incoming data) and FOCM(BB)= Fan-out complexity	component coupling complexity metric for black box componentt CCCM(BB)
8	[27]			Component complexity metric

III. METHODOLOGY

To identify the reusability of components, we define the 'Reusability-matrix' using Reusability-metric. We also proposed, two levels of Reusability-metrics: Individual Component level and CCBS System level. For Reusability-metric we define 3 types of Line of Code of a component:
IndependentLOC(i.eLOCcIndpt),ReusableLOC(i.eLOCcReusab),TailoredLOC(i.eLOCcTailrd),TotalLOC(i.eLOCcTotal).Use

Total LOC (LOCcTotal): It is defined as the summation of the used line of code and independent line of code.

$$\|LOCcTotal - i\| \stackrel{def}{=} \|LOCcReusab - i\| + \|LOCcIndpt - i\| \quad \textbf{(i)}$$

ReusableLOC(LOCcReusab): ReuseableLOCs comprise of ready-to-useLOC and TailoredLOCs of a component.

$$\|\mathbf{LOCcReusab - i}\| \stackrel{def}{=} \|\mathbf{LOCcready - to - use - i}\| + \|\mathbf{LOCcTailrd - i}\| \quad \textbf{(ii)}$$

TailoredLOC (LOCcTailrd): TailoredLOCs consist of major and minor customized LOCs. TailoredLOCs come under the category of reusableLOCs but cannot be reused exactly. TailoredLOCs require fully or partially changes in the code.

$$\|\mathbf{LOCcTailrd - i}\| \stackrel{def}{=} \|\mathbf{LOCcfully - tailrd - i}\| + \|\mathbf{LOCcpartially - tailrd - i}\| \quad \textbf{(iii)}$$

IndependentLOC (LOCcIndpt): IndependentLOC of a component is a difference of TotalLOC and ReusableLOC.

$$\|\mathbf{LOCcIndpt - i}\| \stackrel{def}{=} \|\mathbf{LOCcTotal - i}\| + \|\mathbf{LOCcReusab - i}\| \quad \textbf{(iv)}$$

(i) Individual Component level Reusability-Metric (IRMCi)

Using Line of codes, Reusability-metric at the individual component level is defined as the ratio between the total number of reused line of code of the component and the total number of Line of codes of that component.

$$\text{IRMCi} = \frac{|\ \text{LOCcReusab} - \text{i}\ |}{|\ \text{LOCcTotal} - \text{i}\ |} \quad \textbf{(v)}$$

(ii) CCBS System level Reusability-Metric (SRMCCCBS)

In this section, Reusability-metric is computed by counting the number of IndependentLOC, partially-tailored LOC, fully-tailored LOC and ready-to-use LOC line of code of the Complete Component-Based Software (i.eCCBS). Here, reusability-metric is defined as the ratio between the total reused line of codes of all the components and the total line of codes of the CCBS application, as:

$$\text{SRMCCCBS} = \frac{|\ \text{LOCcReusab}\ |}{|\ \text{LOCcTotal}\ |} \quad \textbf{(vi)}$$

In this paper, LOC Reusability-metric for the following category of components is defined:

- ***Tailored components, which includes fully-tailored and partially-tailored components.***
- ***Ready-to-use components, which can be reused without any modification.***

Tailored Reusability-Metric (RMcTailored) :

Tailored components are the components which can put up with existing requirements, design, code, or test cases with slight or key modifications. On the basis of the degree of modifications, tailored components are divided into two categories:

A. Fully-tailoredcomponents, and

B. Partially-tailoredcomponents.

Fully-Tailored components are the components which require no modification or a slight degree of modification, and partially-tailord components are the components which require a key degree of modification.

On basis of the degree of modifications, we have two conditions in each category (fully-tailored components, and partially-tailored components):

Condition 1: Individual Component level tailored Reusability-metric, and
Condition 2: CCBS system level tailored Reusability-metric.

In this paper, reusability assessment in both the terms is defined, that is, in terms of the line of codes of individual components as well as in terms of a line of codes of system level.

Fully-Tailored Components:

Condition 1: Individual Component Level fully tailored Reusability-Metric (IRMCi):

Fully-tailored components require a slight modification to fit into the new contexts. At the component level, reusability is assessed in terms of individual component LOC. Reusability-metric is described for fully-tailored components in three different contexts:

(i) When complete line of codes of component are involved
It is the condition when the ratio of fully-tailored LOC of a particular component Ci is taken with the total number of lines of the code of that component. This is defined as:

$$\mathbf{IRMCcfully - tailrd - i} = \frac{|\ \mathbf{LOCcfully - tailrd - i}\ |}{|\ \mathbf{LOCcTotal - i}\ |} \quad \textbf{(vii)}$$

(ii) When only reusable line of codes of component are involved :
It is the condition when the ratio of fully-tailoredLOC of a particular component Ci is taken with the total number of reusable line of code of that component. This is defined as:

$$\mathbf{IRMCcfully - tailrd - i} = \frac{|\ \mathbf{LOCcfully - tailrd - i}\ |}{|\ \mathbf{LOCcReusab - i}\ |} \quad \textbf{(viii)}$$

Where, a total reusable line of code $(\text{LOCcReusab} - \text{i})$ represents the collection of ready -to –use LOC and tailored LOC of an individual component ,from Equation (ii).

(iii) When only tailored line of codes of component are involved
It is the condition when theratio of fully-tailoredLOC of a particular component Ci is taken with the total number of reusable line of code of that component. This is defined as:

$$\mathbf{IRMCcfully - tailrd - i} = \frac{|\ \mathbf{LOCcfully - Tailrd - i}\ |}{|\ \mathbf{LOCcTailrd - i}\ |} \quad \textbf{(ix)}$$

where, a tailored line of code $(LOCcTailrd - i)$ represents the collection of fully-tailoredLOC and partially-tailoredLOC of an individual component ,from Equation (iii).

Condition 2: CCBS system level fully tailored Reusability-metric.Individual(RMCCCBS-fully-Tailored):

We can also define tailored Reusability-metric in the following three different contexts in terms of count of components:

(i) when the total line of codes count of components are considered

In this case, the ratio is taken in the context of a total number of components participating in the CCBS system. Here, the total numbers of fully-tailored reused components are divided by the total numbers of components involved in the CCBS development.

$$\mathbf{SRMCCCBS - fully - tailrd} = \frac{|\mathbf{LOCcfully-tailrd}|}{|\mathbf{LOCcTotal}|} \quad \textbf{(x)}$$

(ii) When only reusable line of codes of components are involved

It is the condition when the ratio of fully-tailoredLOC of a particular component Ci is taken with the total number of reusable line of code of that component. This is defined as:

$$\mathbf{SRMCCCBS - fully - tailrd} = \frac{|\mathbf{LOCcfully-tailrd}|}{|\mathbf{LOCcReusab}|} \quad \textbf{(xi)}$$

where, a total reusable line of code $(LOCcReusab)$ represents the collection of ready -to -useLOC and tailoredLOC of components, from Equation (ii).

(iii) When only tailored line of codes of components are involved

It is the condition when the ratio of fully-tailoredLOCof components C is taken with the total number of reusable line of code of all components of thesystem. This is defined as:

$$\mathbf{SRMCCCBS - fully - tailrd} = \frac{|\mathbf{LOCcfully-tailrd}|}{|\mathbf{LOCcTailrd}|} \quad \textbf{(xii)}$$

where, a tailored line of code $(LOCcTailrd)$ represents the collection of fully-tailoredLOC and partially- tailoredLOC of all components, from Equation (iii).

Partially-Tailored Components:

Condition 1: Individual Component Level partially tailored Reusability-Metric (IRMCpartially-tailored-i):

Partially-tailored components require a key degree of modification to fit into the new contexts. At the component level, reusability is assessed in terms of individual component LOC. Reusability-metric is described for partially-tailored components in three different contexts:

(i) When complete line of codes of component are involved

It is the condition when theratio of partially-tailoredLOC of a particular component Ci is taken with the total number of line of code of that component. This is defined as:

$$\mathbf{IRMCcpartially - tailrd - i} = \frac{|\mathbf{LOCcpartially-tailrd-i}|}{|\mathbf{LOCcTotal-i}|} \quad \textbf{(xiii)}$$

(ii) When only reusable line of codes of component are involved

It is the condition when theratio of partially-tailoredLOC of a particular component Ci is taken with the total number of reusable line of code of that component. This is defined as:

$$\mathbf{IRMCcpartially - Tailrd - i} = \frac{|\mathbf{LOCcpartially-tailrd-i}|}{|\mathbf{LOCcReusab-i}|} \quad \textbf{(xiv)}$$

where, a total reusable line of code $(LOCcReusab - i)$ represents the collection of ready -to -useLOC and tailoredLOC of an individual component,from Equation (ii).

(iii) When only tailored line of codes of component are involved

It is the condition when the ratio of partially-tailoredLOC of a particular component Ci is taken with the total number of reusable line of code of that component. This is defined as:

$$\mathbf{IRMCcpartially - tailrd - i} = \frac{|\mathbf{LOCcpartially-tailrd-i}|}{|\mathbf{LOCctailrd-i}|} \quad \textbf{(xv)}$$

Where,a tailored line of code $(LOCcTailrd - i)$ represents the collection of fully-tailoredLOC and partially-tailoredLOC of an individual component ,from Equation (iii).

Condition 2: CCBS system level fully tailored Reusability-metric.Individual (SRMCCCBS):

We can also define tailored Reusability-metric in the following three different contexts in terms of count of components:

(i) In terms of total line of code count of components

In this case, the ratio is taken in the context of a total number of components participating in the CCBS system. Here, the total numbers of partially-tailored reused components are divided by the total numbers of line of code involved in the CCBS development.

$$\mathbf{SRMCCCBS - partially - tailord} = \frac{|\mathbf{LOCcpartially-tailrd}|}{|\mathbf{LOCcTotal}|} \quad \text{(xvi)}$$

(ii) When only reusable line of codes of components are involved

It is the condition when the ratio of partially-tailoredLOC of a particular component Ci is taken with the total number of reusable line of code of that component. This is defined as:

$$\mathbf{SRMCCCBS - partially - tailord} = \frac{|\mathbf{LOCcpartially - tailrd}|}{|\mathbf{LOCcReusab}|} \quad \text{(xvii)}$$

Where, a total reusable line of code $(LOCcReusab)$ represents the collection of ready -to -useLOC and tailoredLOC of components, from Equation (ii).

(iii) When only tailored line of codes of components are involved

It is the condition when the ratio of partially-tailoredLOCof components C is taken with the total number of reusable line of code of all components of the system. This is defined as:

$$\mathbf{SRMCCCBS - partially - tailord} = \frac{|\mathbf{LOCcpartially - tailrd}|}{|\mathbf{LOCcTailrd}|} \quad \text{(xviii)}$$

where, a tailored line of code $(LOCcTailrd)$ represents the collection of partially-tailoredLOC and partially-tailoredLOC of all components, from Equation (iii).

Experimental Analysis of Proposed Metrics

Suppose we have six components for the same task and have to select one component among the six components in such as a manner that it reduce the effort, cost and complexity of software. Six components along with corresponding LOCs given in Table 3

Table 3

Line Of Code	C1	C2	C3	C4	C5	C6
LOCcTotal-i	600	620	580	570	480	600
LOCcfully-tailrd-i	140	220	100	300	140	300
LOCcpartially-tailrd-i	300	230	280	120	200	140
LOCcready-to-use-i	100	90	130	60	50	110

Now using Table 3 and Equation ii, Equation iii, Equation iv we can calculate Reusable (LOCcReusab), Tailored (LOCcTailrd) and independent(LOCcIndpt) line of codes of the candidate components.

Table 4 Reusable,Tailored and Independent line of codes of 6 Candidate Components

Line Of Codes	C1	C2	C3	C4	C5	C6
LOCcReusab	540	540	510	480	390	440
LOCcTailrd	440	450	380	420	340	340
LOCcIndpt	60	80	70	90	90	70

With the help of Table 3 and Table 4 now we can draw the Reusability-matrix for six candidate components using the Reusability-metric method.

Case 1: Reusability-matrix when all parts of the components are involved in the computation.

Applying the values given in Table 3 and Table 4 on Equation (v), Equation (xvii), Equation (viii) respectively, we assess the values of Reusability-metrics of component Ci, in the context when calculations are made in terms of all parts of the component, that is, Independent and Reusable. These computations are shown in Table5

Table 5 Reusability-Matrix when Independent and the reusable Line of Codes are involved

Line Of Code	C1	C2	C3	C4	C5	C6
IRMCcTotal-i	0.9	0.87	0.88	0.84	0.81	0.5
IRMCcfully-tailrd-i	0.23	0.35	0.17	0.53	0.29	0.17
IRMCcpartially-tailrd-i	0.5	0.37	0.48	0.21	0.42	0.9
IRMCcready-to-use-i	0.17	0.14	0.22	0.1	0.1	0.23

Case 2: Reusability-matrix when theonly Reusable line of codes is involved.

Below table depicting the Reusability-Matrix when the only Reusable line of codes of the particular Component.

Table 6 Reusability-Matrix when only Reusable Line of codes of the Component is involved

Reusablility Matrix	C1	C2	C3	C4	C5	C6
LOCcpartially-tailrd-i	0.26	0.41	0.20	0.62	0.36	0.17
LOCcfully-tailrd-i	0.55	0.42	0.55	0.25	0.51	0.42
LOCcready-to –use	0.18	0.17	0.25	0.12	0.13	0.13

Case 3: Reusability-matrix when the only Adaptable line of codes of components is involved.

Applying the values given in Table 3 and Table 4 on the above given respective equation, we calculate the values of Reusability-metrics of component Ci, in the context of adaptable function points only.

Table 7 Reusability-matrix when the only Adaptable line of codes of components is involved.

Reusablility Matrix	C1	C2	C3	C4	C5	C6
LOCcpartially-tailrd-i	0.32	0.49	0.26	0.71	0.41	0.31
LOCcfully-tailrd-i	0.68	0.51	0.74	0.29	0.59	0.32

IV. RESULT

Selection and Verification of Candidate Components

After calculating the values of reusability-matrix we now define the selection procedure of the candidate components. As from reusability-matrix shown in Table 3, Table 4, and Table 5, we note that each row contains the value of line of code of the corresponding candidate components. For example, the first row of Table 3 contains the values of Reusability-metric of component Ci (IRMCi) of corresponding candidate components from C1 to C6. We also note that the component C1 have the maximum Reusability-metric value in the perspective of the total component. We select the component C1 when reusability is considered in the context of all parts of the component.

From the Reusability-matrix computed in Tables 3, Table 4 and Table 5, we can select and verify the selection of components.

Selection of components when all the parts of the component are considered:

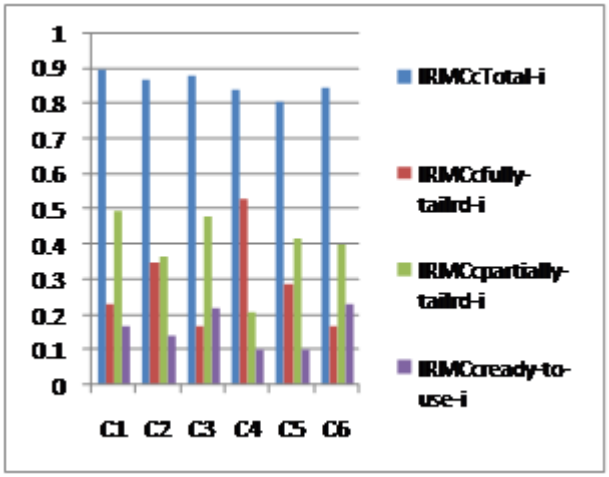

Figure 1 Reusability-Graph when the Components containing New and Reusable Line of code.

From Table 3 and Figure 1 we note that, when selection is made at the component level, then the eligible components are:

(a) Highest Reusability-metric value: Component C1

(b) Highest partially-qualified component: Component C4

(c) Highest fully-qualified component: Component C3

(d) Highest off-the-shelf component: Component C3

Selection of components when Reusable components are considered:

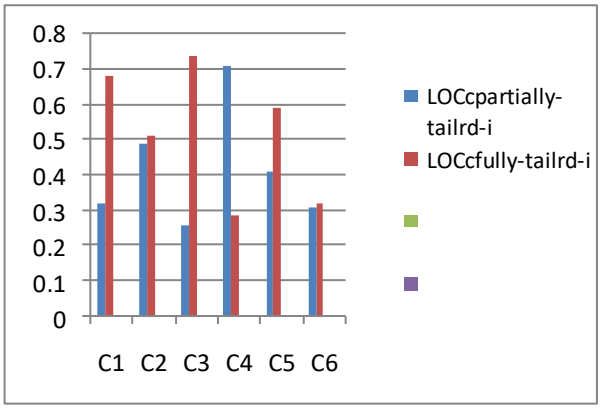

Figure2 Reusability-Graph when the Components containing Reusable line of code only

From Table 4 and Figure 2, we note that when selection is made at the reusable component level, then the eligible components are:

(a) Highest partially-tailored component: Component C14

(b) Highest fully-tailored component: Component C11 and Component C13

(c) Highest ready-to-use component: Component C13

Selection of components when only tailored components are considered:

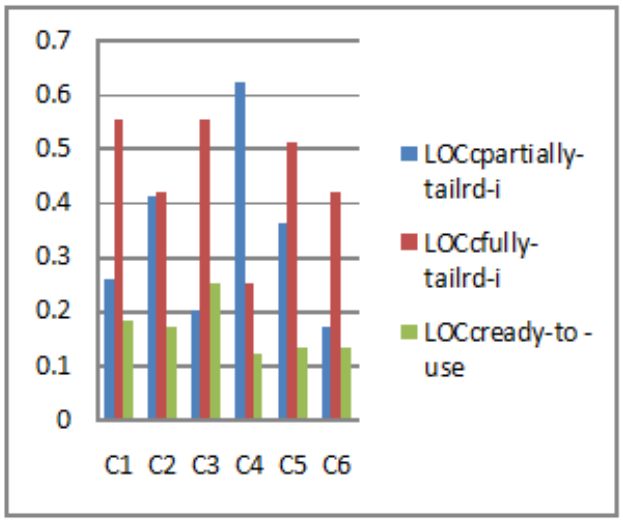

Figure 3 Reusability-Graph when the Components containing Tailored Line of code only

From Table v and Figure 3, we note that when selection is made at the adaptable component level, then the eligible components are:

(a) Highest partially-qualified component: Component C12,

(b) Highest fully-qualified component: Component C11 and Component C13,

We can select components according to our criteria and selection level. Selection process of components at system level can be defined in similar manner. At System level we store all the values in terms of applications, and all the computations are in terms of function points defined at CBS System level.

Validation of above proposed metrics:

In order to validate the proposed metric, the other metrics named Self-Completeness of Component's Parameter (SCCp) and Self-Completeness of Component's Return Value (SCCr) defined by Washizaki et al. [28][29] have been used. SCCp and SCCr metrics are used in determining the external dependency and portability of Components. "In business methods, there is a possibility that parameters and return values depend on the rest of the software which originally use the component"[28].

Karl Pearson's Correlation Coefficient is calculated between IRMCcTotal-i and SCCp, and between IRMCcTotal-iand SCCr in order to validate and correlate the proposed metric with quality characteristic reusability.

Table8:The values of SCCp and SCCr metrics for each components

Component Name	SCCp	SCCr
C1	0.667	0.667
C2	0.5	0.5
C3	0.5	0.5
C4	0.5	0.5
C5	0.75	1
C6	0.75	1

Validation of the Proposed Complexity Metric

A correlation analysis has been carried out for the proposed reusability complexity metric IRMCcTotal-i with the metrics SCCp and SCCr by using the Karl Pearson Coefficient of Correlation. The formula for calculating the Karl Pearson Correlation Coefficient is as below:

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

The following Table 8 shows the calculations for Karl Pearson Correlation Coefficient between IRMCcTotal-i and SCCp.

Table- 9: Calculations for the Karl Pearson Correlation Coefficient between IRMCcTotal-i , SCCp and SCCr

Component Name	X= IRMCcTotal-i	Y= SCCp	X_1^2	Y_1^2	XY	Y= SCCr	X_2^2	Y_2^2	XY
C1	0.9	0.667	.81	.445	.6003	0.667	.81	0.445	.6003
C2	0.87	0.5	.7569	.25	.435	0.5	.7569	0.25	.435
C3	0.88	0.5	.7744	.25	.44	0.5	.7744	0.25	.44
C4	0.84	0.5	.7056	.25	.42	0.5	.7056	0.25	.42
C5	0.81	1	.6561	1	.81	0.75	.6561	0.563	.6075
C6	0.5	1	.25	1	.5	0.75	.25	0.563	.375
	∑X = 4.8	∑Y= 4.167	$\sum X^2$ =3.953	$\sum Y^2$=3.195	∑XY=3.2053	∑Y= 3.667	∑X2 =3.953	∑Y2=2.321	∑XY=2.8778

V. CONCLUSION

In CBSE components component connected and communicated each other on the basis of parameters, services and functionalities. In this paper we define a quantifiable method to measure and analyses the reusability of the component in CBSE. We categories the components into three classes: independent LOC, reusable LOC, total LOC. We propose a methodology of the best selection of reusable component from an existing repository. To show our findings we have use graph-based method and find the best suitable component. In future we have scope to find the method for estimation for the risk associated with the component reuse.

REFERENCES

1. Yahya Tashtoush1, Mohammed Al-Maolegi, Bassam Arkok, Correlation among Software Complexity Metrics with Case Study, International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume-4 Number-2 (2015
2. Jianguo Chen, Hui Wang, Yongxia Zhou, Stefan D. Bruda : "Complexity Metrics for Component-based Software Systems" in International Journal of Digital Content Technology and its Applications. Volume 5, Number 3, March 2011
3. Alireza Sadeghi, Naeem Esfahani, and Sam Malek: " Ensuring the consistency of adaptation through inter- and intra-component dependency analysis." in Transactions on Software Engineering and Methodology, Vol. 26, No. 1, Article 2, May 2017.
4. Sheng Yu, Shijie: A Survey on Metric of Software Complexity" 978-1-4244-5265-1/10/ in IEEE, 2010
5. Usha Kumari, Sucheta Bhasin: "A Composite Complexity Measure For Component-Based Systems: in ACM SIGSOFT Software Engineering Notes Page 1 Volume 36 Number 6, Nov 2011
6. Timothy M. Meyers And David Binkley: "An Empirical Study of Slice-Based Cohesion and Coupling Metrics" in ACM Transactions on Software Engineering and Methodology, Vol.17, No.1, Article2, Dec 2007.
7. Jianguo Chen, Wai K. YEAP, Stefan D. Bruda: "A Review of Component Coupling Metrics for Component-Based Development " in World Congress on Software Engineering, IEEE Conference Publications,2009
8. Chuanqi Tao, Jerry Gao, Bixin Li: " AModel-Based Framework to Support Complexity Analysis Service for Regression Testing of Component-Based Software", in IEEE Symposium on Service-Oriented System Engineering, 2015
9. Ratneshwer,A K Tripathi: "Dependence Analysis of Software Component" in ACM SIGSOFT Software Engineering Notes Page 2 Volume 35 Number 4, July 2010.
10. Umesh Tiwari and Santosh Kumar: "In-Out Interaction Complexity Metrics for Component-Based Software", ACM SIGSOFT Software Engineering Notes, vol. 39, no. 5, September 2014.
11. Slyngstad N. P. O., Gupta A., Conradi R., Mohagheghi P., Ronneberg H., and Landre F: "An Empirical Study of Developers Views on Software Reuse in Statoil ASA", in Jose Carlos Maldonado and ClaesWohlin (Eds.), Proc. 5th ACM-IEEE Int'l Symposium on Empirical Software Engineering(ISESE'06), IEEE CS Press, ISBN 1-59593-218-6, pp. 242-251, 21-22 , September 2006.
12. Szyperski, C. 1997. Component Software: Beyond ObjectOriented Programming. Addison-Wesley, (1997).
13. Schneider, J. G., and Nierstrasz, O. 1999. Script Components and glue. Software Architectures Advances and Applications, Springer, 13-15, (1999). Shepperd, M. 1988. A critique of Cyclomatic complexity as software metric. Software Engineering Journal, 3036, (March 1988)M. King, B. Zhu, and S. Tang, "Optimal path planning," *Mobile Robots*, vol. 8, no. 2, pp. 520-531, March 2001.
14. Prieto-Diaz, Ruben and Peter Freeman, "Classifying Software for Reusability", IEEE Software, vol.4, no.1, pp.6-16, January 1987
15. Chen, Deng-Jyi and P.J. Lee, "On the Study of Software Reuse Using Reusable C++ Components", Journal of Systems Software, vol.20, no.1, pp. 19-36, Jan 1993.
16. G. Caldiera and V.R. Basili, "Identifying and qualifying reusable software components", IEEE Computer, vol. 24, Feb. 1991.
17. Jeffrey S. Poulin, "Measuring Software Reusability", In Proc. Third International Conference on Software, Rio de Janerio, Brazil, 1-4 Nov. 1994, pp. 126-138.
18. W. Gregory Hislop, "Using Existing Software in a Software Reuse Initiative", The Sixth Annual Workshop on Software Reuse (WISR'93), Owego, New York, 2-4 November 1993.

19. J. Barnard, "A new reusability metric for object-oriented software", Software Quality Journal, vol. 7, pp. 35-50, Jan. 1998.
20. Gill, N, Grover P. Component-Based Measurement: Few Useful Guidelines. ACM SIGSOFT Software Engineering Notes, vol. 28, 2003.
21. Chen, Wai K. YEAP, Stefan D. Bruda,A Review of Component Coupling Metrics for Component-Based Development Jianguo
22. Latika Kharb, Rajender Singh, Complexity Metrics for Component-Oriented Software Systems, ACM SIGSOFT Software Engineering Notes Page 2 March 2008
23. Nasib S. Gill and Balkishan,Dependency and Interaction Oriented Complexity Metrics of Component-Based Systems, ACM SIGSOFT Software Engineering Notes Page 1 January 2008
24. V. L. Narasimhan and B. Hendradjaya.,A New Suite of Metrics for the Integration of Software Components
25. U.Kumari and S.Upadhyaya, An interface complexity measure for component-based software systems, International Journal of Computer Applications, 36(1) (2012) 4652.
26. P.Trivedi and R.Kumar, Software metrics to estimate software quality using software component reusability, International Journal of Computer Science, 9(2) (2012)144-149
27. N.Kaur and A.Singh, A complexity metric for black box components, International Journal of Soft Computing and Engineering, 3(2) (2013) 179-184. 67. V.P.Venkatesan and M.Krishnamoorthy, A metrics for measuring software components, JCIT, 4(2) (2009) 138-153.
28. Washizaki, Hironori & Yamamoto, H & Fukazawa, Yoshiaki. (2003). A metrics suite for measuring reusability of software components. Proc. 9th IEEE International Symposium on Software Metrics. 211-223.
29. Washizaki H., Hiraguchi H., Fukazawa Y. (2008) A Metrics Suite for Measuring Quality Characteristics of JavaBeans Components. In: Jedlitschka A., Salo O. (eds) Product-Focused Software Process Improvement. PROFES 2008. Lecture Notes in Computer Science, vol 5089. Springer, Berlin, Heidelberg

AUTHORS PROFILE

Nidhi Mehra, MCA, M.Teh Software Engineering, Green Cloud Computing

Divya Kapil, MSc(Maths), MCA, MTech(CSE), PHD(P), Cloud Computing, Machine Leaning.

1. D. Kapil, P. Tyagi, S. Kumar and V. P. Tamta, "Cloud Computing: Overview and Research Issues," 2017 International Conference on Green Informatics (ICGI), Fuzhou, 2017, pp. 71-76.
2. Divya Kapil, Emmanuel S. Pilli and Ramesh C. Joshi, "Live Virtual Machine Migration Techniques: Survey and Research challenges", 3rd IEEE International Advance Computing Conference (IACC), 2013.
3. Parshant Tyagi , Divya Kapil, , Emmanuel S. Pilli and Ramesh C. Joshi, "Virtual Machine Portability- A Novel Approach", CloudCom IEEE, 2013.
4. Divya Kapil, "Live Virtual Machine migration: A Performance Test Xen and KVM", IJRTE, 2020