Evaluating Quality of Software Component using Metrics

Gagan Prakash Negi
Department of Computer Science, Abhilashi University Mandi H.P.
ambition.negi@gmail.com

Abstract: - *The main aim of software engineering community is to provide a high quality system or a product. To achieve this in recent years, many considerable efforts have put by software engineers in to the design and development of Component based software system (CBSS).This Paper presents some such efforts by using measurement tools and technique, i.e. through the effective software metrics. We provide an account of novel software measures for component by adequate coupling, cohesion and interface metrics. The advantages of our technique are discussed as well through a case study in this paper.*

Introduction

Component-based software (CBS) construction has brought a new revolution in software engineering. Component based software engineering is a methodology that emphasizes the design and construction of computer-based systems using reusable component. It's also developing a kind of software which relies on reuse and it emerged from the failure of object oriented development. Component provides system functionality by interacting, cooperation and coordinating. Interaction, Cooperation will produce dependencies among them. Usually, groups of component depend on each other in order to supply complex system functionality. Any modification to a component can cause the change of composite functionality, because the composite functionality is reflected in different component. In addition, the replacement of a new version component will cause the change of dependency between components.

Software organizations invest some how 80% of their development resources for issues related to their product quality. In CBS, quality aspect becomes more important due to its architectural difference. Here, application developers have to rely on the component vendors or the developers. The quality of the component will have a very high impact on the quality of the final system. The quality of the application is high when it yields the expected results, is stable and adaptable and leads to reduce the maintenance costs. Metrics are needed to measure, is stable and adaptable and leads to reduce the maintenance costs. Metrics are needed to measure several types of quality issues and to study the characteristics of a given software system under different scenarios. CBSE metrics required to evaluate the behavior and reliability of the component when integrated in to large software system. Metrics that have a sound theoretical basis become applicable to real life organizations. Some of the metrics reply on the parameters that could never be measure or are too difficult to measure in practice.

Due to the inherent difference in the development of component based and non-component based systems, the traditional software metrics prove to be inappropriate for component –based systems. The component metrics alone are not sufficient for an integrated environment, because there is need to measure the stability and adaptability of each component when it is integrated with other components. There is need to measure the stability and adaptability of each component when it is integrated with other components. There is need of metrics to assess the functionality of each component, when integrated with other component and functionality of the application. Quality assurance was relatively unchallenged tasks. In the last few years this situation has changed. Some quality models have been widely adopted to significantly improve quality factors such as scalability, flexibility, functionality and availability. The rise of new models has introduced estimation of quality factor in the component based software system application domain with versatile challenges.

COMPONENT BASED SOFTWARE METRICS

Component Based development is an approach in which a system is developed from the well defined, independently produced pieces known as components. A software component is a unit of packaging, distribution or delivery that provides services within a data integrity or encapsulation boundary. It is a unit of compositions with contractually specified interface and explicit context dependencies only. It can be placed on any node in depending on application. Needs and regardless on the type of particular Network structure. An extra effort must be paid for the additional functionality of the component beyond the current applications need, to make the component more useful. It is black box entities that encapsulate service behind well defined interfaces. Components are not used in isolation; they may also have the rules of governing their composition.

IEEE stated that measure is a quantitative indication of extent, amount, dimensions, capacity, or size of some attribute of a product or process. The IEEE Standard defines metric as, "a quantitative measure of degree to which a system, component or process posses a given attribute. An indicator is a metric or combination of metrics that provides in sight in to the software process, a software project, or the product itself.

1 Software Metrics

Metric is a quantitative discipline by its nature. Engineers use numbers to help them design and assess the product to be built. It helps software engineers gain insight in to the design and construction of the software they build. It focuses on specific attributes of software engineering work products and are collected as technical tasks (analysis, design, coding, and testing) are being conducted. Software engineers use product metrics to help them build higher-quality software. There will always be a qualitative element to the creation of computer software. The problem is that quantitative assessment may not be enough. A software engineer needs objective criteria to help guide the design of data, architecture, interface, and components. The tester needs quantitative guidance that willhelp in the selection of test cases and their targets. Metrics provide a basis from which analysis, desire objectively and assessed more quantitatively.

The IEEE Standards states that a measure provides a quantitative indication of the extent, amount, dimensions, capacity, or size of some attribute of a product or process. Measurement is the act of determining a measure. The IEEE Standards defines metric as, a quantitative measure of degree to which a system, component or process posses a given attribute. "A handle or guess about a given attribute". Measurement occurs as the result of the collection of one or more data points. A software engineer collects measures and develops metrics so that indicators will be obtained .The IEEE standards defines indicator as, a metric or combination of metrics that provides insight into the software process, a software project, or the product itself. It provides insight that enables the software engineers to adjust the process, the project, or the product to make things better.

2 Categorization of Component Based Metrics

Software component is usually regarded as a part of the starting platform for service orientation throughout software engineering. It can be regarded as reusable software element such as a function, file, class, module or subsystem. In CBSS, each acquired component should be validated at a very early stage in the software development process such as the design stage and coding one. The quality attributes of the component must be measured using validated metrics that should provide a correct estimate of the attributes. Here we will mainly discuss component metrics and system measures at the design stage of life cycle.

2.1 Component general metrics

The important and relevant metrics applicable for the component quality analysis during design stages are:

Component Size Metric (CSM): CSM should be based on the total number of sub-components such as classes or use cases.

Weighted Methods per Component (WMC): The number of local methods defined in the component, WMC is related to size complexity. WMC is the indicator of development and maintainability complexity.**Depth of Inheritance Tree (DIT):** The maximum depth of the component in the inheritance tree. The deeper the components are in the inheritance hierarchy, the greater the number of methods it is likely to inherit, making it more complex to predict the component's behavior.

Number of Children (NOC): The number of immediate sub-components of a component or the count of derived components, NOC measure inheritance complexity

Count of Base Components (CBC): The number of base components likes NOC, CBC measures inheritance complexity

Response Set For a Class (RFC): The set of methods that can potentially be executed in response to a message received by an object of that component .RFC is simply the number of methods in the set, including inherited methods.

Additionally, we will present three more metrics for the component in CBSS: coupling, cohesion and interface metrics. Cohesion is a measure of how strongly-related the various responsibilities of a software component are. Coupling is usually contrasted with cohesion. Low coupling often correlates with high cohesion, and vice versa.Interface metric providesan estimate of thecomplexity of interfaces.

2.2 Coupling Metric

Coupling between components is the number of other components coupled to this component. In CBSS, coupling will be defined as: two components are coupled if and only if at least one of them acts upon the other. Since coupling is the extent to which the components are interdependent, a quantitative measure is to count the way in which one component may dependent on the other. In order to develop a coupling metric, we begin by regarding any CBSS as a directed graph. The components comprising CBSS are the vertices in the graph. Suppose such a system comprises a set of components C ≡ {C1, C2,…Cm}. Let Mj and Vj be the sets of methods and instance variables of component Cj. MVj,i is the set of methods and instance variables in class Ci invoked by class Cj. MVj, the set of all methods and instance variables in other components, $1 \leq i \leq m$ and $i \neq j$, that are invoked by component Cj,

$$MVj = \frac{|\cup MVj,i|}{|Mj|+|Vj|}$$

There are usually two kinds of coupling: afferent coupling and efferent coupling. Afferent coupling is the number of other components that depend upon sub-components within the component and is an indicator of the component's responsibility. Efferent coupling is the number of other components that the sub-components in the component depend upon and is an indicator of the component's independence.

2.3 Cohesion Metric

Cohesion specifies the similarity of methods in a component. It is a measure of the extent to which the various functions performed by a component are related to one another. Suppose a component j such as a class has a set ofmethod members Mj(C) ≡ { mj1, mj2,…mjm} and a set of instance variables Vj (C) ≡{vj1, vj2, …. vjn}. Ej (C) is the set of

$$COMj(C) = \frac{|\ Ej(c)|}{|Vj(c)| * |Mj(c)|}$$

pairs (vj,mj) for each instance variable v in V(C) that is used by method m in Mj (C). The cohesion metric for a component j, also named as Cohesion of Methods (COM), is defined by above given expression.

2.4 Interface Metric:-

Afferent coupling and efferent coupling can also be represented by component interface metric (CIM). CIM is defined as three values:

- Available number of incoming interactions (II).
- Available number of outgoing interactions (OI).
- A ratio of II to OI, i.e., II/OI.

Another interface metric related to a component j is actual interactions metric (AIM) that measure the interface density in a component .AIM is the ratio between the actual numbers of interactions over potential ones:

$$AIMj = (IIj + OIj) / (IIj, max + OI\ j, max)$$

Here IIj,max and *OI j, max* are maximum numbers of input and output interactions in a component j. We will use AIM instead of CIM as the interface metric for component.

2.5 Sole Component Complexity Metric

If a sole component complexity metric (SCCM) is required, we may combine above three component metrics with different weights for each metric.

$$SCCMj = \alpha * MVj + \beta * COMj + \gamma * AIM$$

Where α, β and γ are the weighs for the coupling metric, cohesion and interface metrics of component j with the condition as α + β + γ = 1.

2.6 Component Assembly Metric

Combining component-level metrics to obtain system level indicators of quality is a challenging issue. Component assembly evaluation can be of two types: qualitative or quantitative. The former can be performed by an expert, but is prone to be subjective since different experts may produce conflicting evaluations and the comparison between evaluations performed on different assemblies is difficult.

On the other hand, the quantitative evaluation can be objective and repeatable. Through the usage of well-defined metrics, a quantitative quality model of CBSS may facilitate the comparison of the results of evaluations performed on different assemblies.

Having established the measures for the strength of coupling between pairs of components, the cohesion degree within the components and the interface metrics, the final step is to use them as a basis for the measure of the total complexity of CBSS. This is readily achieved by summing all the measures and dividing by the total number of the components in CBSS.

System coupling metric:- The system coupling metric (SCOUP) for CBSS will be:

$$SCOUP = \frac{\sum_{J=1}^{M} MVj}{\text{m}}$$

Here MVj is the coupling metric for component j and m is the number of the components in CBSS.

System cohesion metric:-The system cohesion metric (SCOH) for CBSS is:

$$SCOH = \frac{\sum_{J=1}^{M} COMj}{\text{m}}$$

Similarly, COMj is the cohesion metric for component j.

System actual interface metric:- Based on the measurement of actual interactions for a component j, system actual interface metric (SAIM) is the integration of the interface metrics of the total number of components:

$$SAIM = \frac{\sum_{J=1}^{M} AIMj}{\text{m}}$$

Sole system complexity metric:- If we need a sole system complexity metric (SSCM), we may combine above three system metrics with different weights for each items.

$$SSCMj = \alpha' * MVj + \beta' * COMj + \gamma' * AIM$$

where α', β' and γ' are the weighs for system coupling metric, cohesion and interface metrics with the condition as α' + β' + γ' = 1.

CASE STUDY: LIBRARY MANAGEMENT SYSTEM

The Library Management System presented here describes some aspects of issuing book to students. Student issuing book pattern is the patterns that include a number of simpler patterns. These patterns describe the creation and maintenance of the student's record and the assignment of the library books for use to student.

1. Reverse Engineering Process to extract information domain of CBSS

Table 7.1 shows the extracted information from the domain of software system using Understand 2.6 tool.

Here,

Mj-CountDeclInstanceMethod	CountInput-IImax
Vj-CountDeclInstanceVariable	CountOutput-IOmax

2. Component Diagram of CBSS.

Edraw Max Tool is used to draw a relationship among three components of the CBSS named as STUDENT, BOOK and ISSUE as shown in Figure 7.1 and Figure 7.2

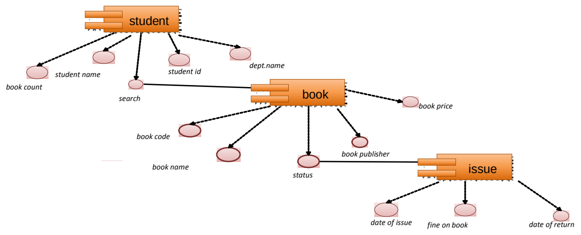


Figure 7.1 Component diagram for Library management system

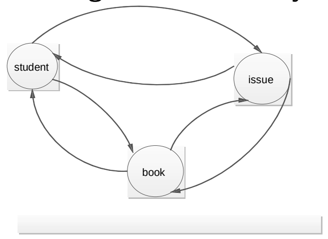


Figure 7.2 Library management systems consisting of three components

3. Computation of Quality Metrics of software component

Table 7.2 indicates the raw complexity data for the various components used to compute

Table 7.2 Raw Complexity data for components

Component		J	Student	Issu e	Book	IImax	OI max	$\|\boldsymbol{Ej(c)}\|$
I	M i	V i	MVj i	MVj i	MVj i			
Student	2	4		2,2	1,1	10	4	7
Issue	5	30	1,2		1,1	47	25	49
Book	3	6	1,1	1,1		16	4	11

various quality metrics as given Table 7.3 shows the coupling metrics for each components these individual metrics are again computed and system coupling of the CBSS is obtain. From (3.2.6), System coupling metric (SCOUP) for the CBSS is 0.53

Table 7.3 Coupling Metrics for component

Compone nt		J	Studen t	Issue	Boo k	MV j
I	M i	V i	MVj i	MVj i	MV j i	
Student	2	4		2,2	1,1	1.00
Issue	5	30	1,2		1,1	0.142
Book	3	6	1,1	1,1		0.444

Table 7.4 is generated after computed$|\boldsymbol{Ej(c)}|$with Mi, Vi according to (3.2.3), Table 7.4 the shows the individual cohesion metric for each component. From (3.2.6), System cohesion metric (SCOH) 0.604 for the system is obtained.

Table 7.4 Cohesion Metrics for component

Component	J	$\|\boldsymbol{Ej(c)}\|$	COMj

I	M i	V i		
Student	2	4	7	0.875
Issue	5	30	49	0.327
Book	3	6	11	0.611

The interface metric for each component in system, shown in Table 7.5.

Table 7.5 Interface metrics for component

Component	**J**		**Student**	**Issue**	**Book**	**IImax**	**OI max**	**II**	**OI**	***AIMj***
I	**M i**	**Vi**	**MVji**	**MVji**	**MVji**					
Student	2	4		2,2	1,1	10	4	3	3	0.429
Issue	5	30	1,2		1,1	47	25	3	3	0.833
Book	3	6	1,1	1,1		16	4	2	2	0.2

Actual interface metric (AIMj) is obtained for each component by (3.2.4), and these each metric helped to compute System actual interface metric (SAIM) for the CBSS i.e. 0.487. Table 7.6 shows the complexity metrics for the components.

Table 7.6 Complexity metrics for components

Component	**MV j**	**COMj**	**AIMj**	**SCCMj**
Student	1.00	0.875	0.429	0.803
Issue	0.142	0.327	0.833	0.386
Book	0.444	0.611	0.2	0.404

Combined sole system complexity metric (SCCM) for the CBSS is obtained from (3.2.6) that is 0.532.

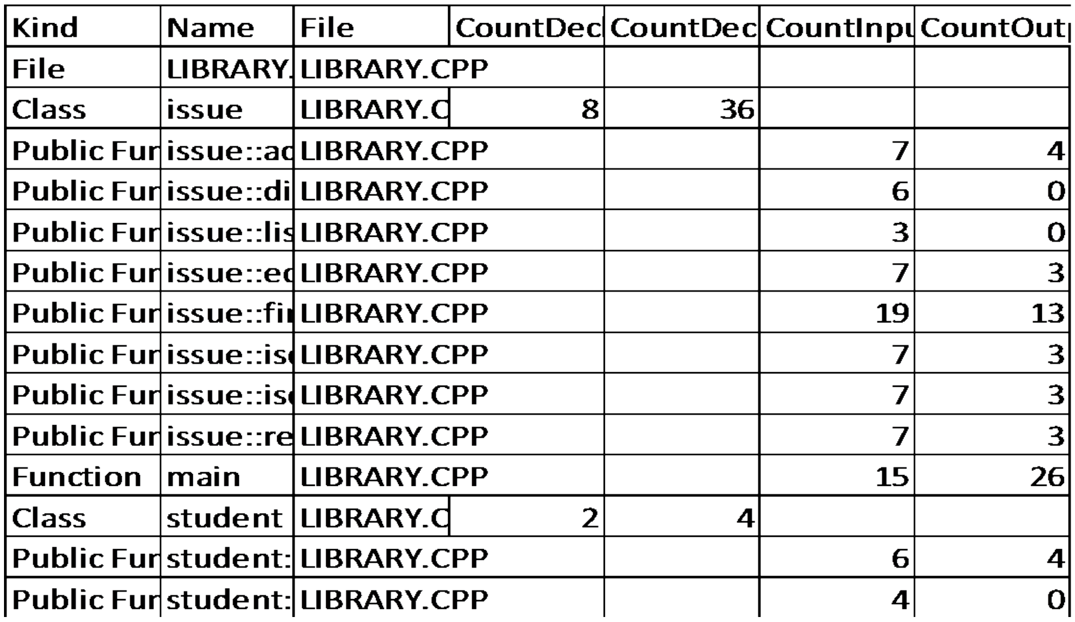
4. Analysis of impact of metrics on quality

From above given table we have seen that the component STUDENT has the highest complexity metric and need the most investment such as time and cost , while the component ISSUE has the lowest complexity value. The complexity metric of component ISSUE is less than that of component BOOK.

5. Refinement of the CBSS

Some changes are made to the design model so as to refine the model and the metrics recomputed. Table7.7 shows the extracted information after redesigning of the CBSS.

Table 7.7 Library management redesigned understand 2.6.csv



Kind	Name	File	CountDec	CountDec	CountInpu	CountOut
File	LIBRARY.	LIBRARY.CPP				
Class	issue	LIBRARY.C	8	36		
Public Fur	issue::ad	LIBRARY.CPP			7	4
Public Fur	issue::di	LIBRARY.CPP			6	0
Public Fur	issue::lis	LIBRARY.CPP			3	0
Public Fur	issue::ed	LIBRARY.CPP			7	3
Public Fur	issue::fi	LIBRARY.CPP			19	13
Public Fur	issue::is	LIBRARY.CPP			7	3
Public Fur	issue::is	LIBRARY.CPP			7	3
Public Fur	issue::re	LIBRARY.CPP			7	3
Function	main	LIBRARY.CPP			15	26
Class	student	LIBRARY.C	2	4		
Public Fur	student:	LIBRARY.CPP			6	4
Public Fur	student:	LIBRARY.CPP			4	0

Figure 7.3 and Figure 7.4 shows the component diagram after refinement, these both figures show the relationship between two components that is STUDENT andISSUEBOOK.

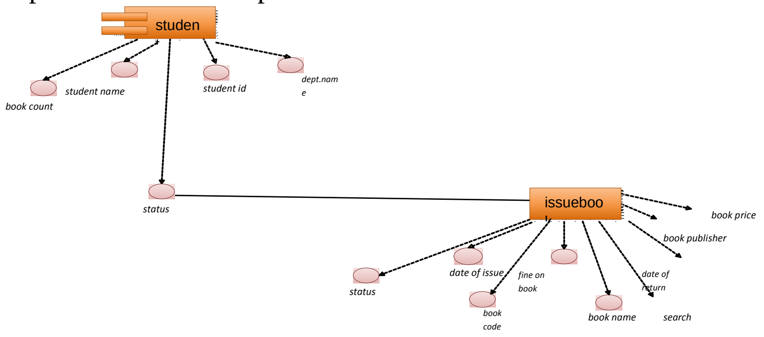


Figure 7.3 Refined Component Diagram for Library Management System

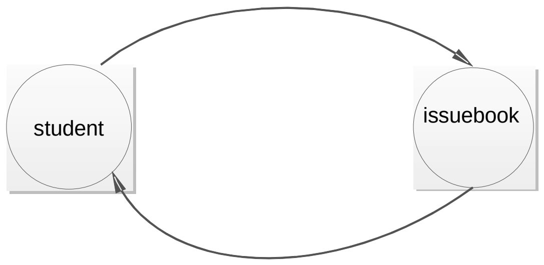


Figure 7.4 Component based diagram of redesigned Library management system.

Table 7.8 shows the recomputed raw complexity data for the components.

Table7.8 Recomputed Raw Complexity data for components

Componen t	**J**		**Student**	**Issue**	**IImax**	**OI max**	**$\|Ej(c)\|$**
I	**M i**	**V i**	**MVj i**	**MVj i**			
Student	2	4		2,2	10	4	7
Issuebook	8	36	1,2		63	29	60

Table 7.9 shows recomputed coupling metrics for the each components and helps to obtain SCOPU i.e. 0.037.

Table 7.9 Recomputed Coupling Metrics for component

Componen t	**J**		**Studen t**	**Issue**	**MV j**
I	**M i**	**V i**	**MVj i**	**MVj i**	
Student	2	4		2,2	0.666
Issuebook	8	36	1,2		0.068

Table 7.10 indicates the cohesion metric for each individual component. These metric used to find SCOH after redesigned the CBSS, i.e. 0.541.

Table 7.10 Recomputed Cohesion Metrics for component

Componen t	**J**		**$\|Ej(c)\|$**	**COM j**
I	**M i**	**V i**		
Student	2	4	7	0.875
Issuebook	8	36	60	0.208

AIMj for each component is shown in Table 7.11, and helped to compute SAIM for CBSS,i.e. 0.164.

Table 7.11 Recomputed Interface metrics for component

Componen t	**J**		**Student**	**Issuebook**	**IImax**	**OI max**	**II**	**OI**	**$AIMj$**
I	**M i**	**V i**	**MVj i**	**MVj i**					
Student	2	4		2,2	10	4	2	2	0.2857
Issuebook	8	36	1,2		63	29	2	2	0.0434

Table 7.12 shows the complexity metric for the component, and used to obtain combined sole system complexity metric SCCM for the CBSS, i.e.0.176

Table 7.12 Recomputed Complexity metrics for the components

Component	**MV j**	**COMj**	**AIMj**	**SCCMj**
Student	0.666	0.875	0.2857	0.594
Issuebook	0.068	0.208	0.0434	0.085

Results and Discussion

A quality component is designed using Edrawl tool. The metric information is extractable from component diagrams using Understand 2.6 tool. Information on each metric and the quality attributes is evaluated and presented. Graphical interpretation of results is implemented in order to facilitate easier metric interpretation. The graphs are expected to be interactive in that sense that since in most case, single columns, points and lines on the graph may represent multiple entities

CASE STUDY: Library Management System

The histograms of few metrics are depicted below. Some background knowledge and suggestion are also given as to how the graphs are to be interpreted.This histogram plots the values for quality metrics. This gives a good overall view to recognize the distribution of coupling,cohesion,actual interface and sole system comboned complexity over the whole project.

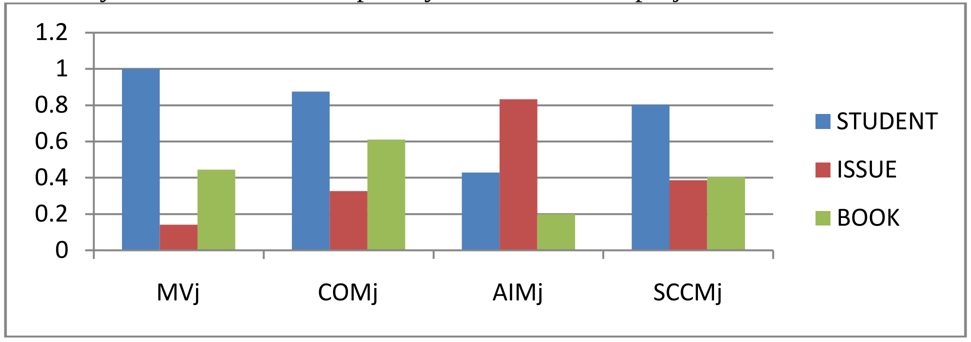


Figure 8.1 Quality Metrics for components

The quality metrics histogram plots the values of MVj, COMj, AIMj and SCCMj against the number of classes. Figure 8.2 shows the distribution of component based metrics for the whole project before refinement.

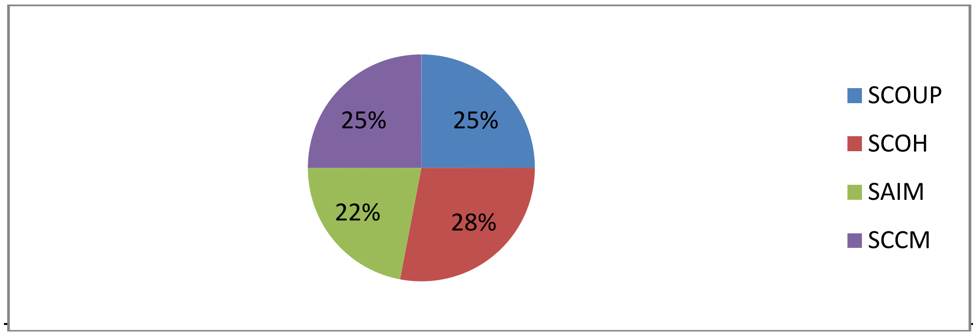


Figure 8.2 CB metrics for CBSS

Before refinement and after refinement the metrics are compared by given histogram in figure 8.3.

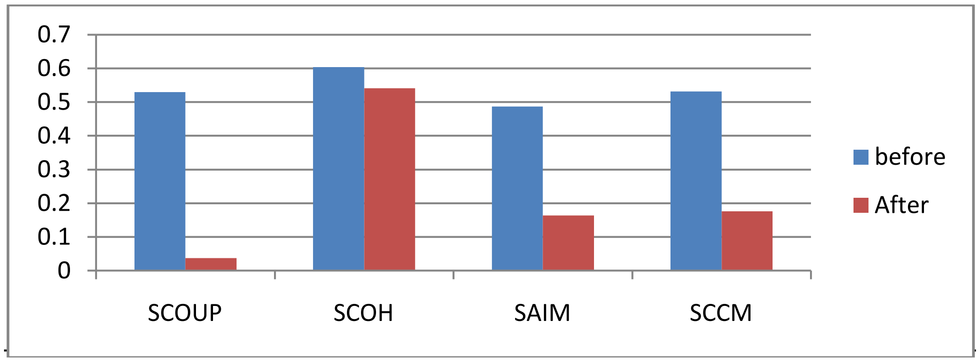


Figure 8.3 Histogram showing comparison between CBSS

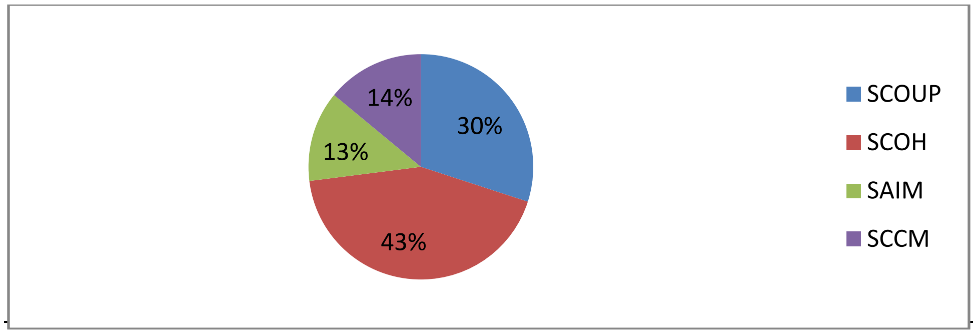


Figure 8.4 CB metrics for CBSS after refinement

Figure 8.4 shows the distribution of component based metrics for the whole project after refinement. System cohesion is higher and least is coupling in the project.

References

[1] Amjan Shaik, C. R. K. Reddy, Bala Manda, Prakashini. C, Deepthi. K, Metrics for Object Oriented Design Software Systems: A Survey, Journal of Emerging Trends in Engineering and Applied Sciences, vol.1, no.2, 2010, pp.190-198.

[2] Bixin L., Managing Dependencies in Component-Based Systems Based on Matrix Model, Proceedings of. Net Object Days, Sep. 2003.

[3] David P. Tegarden, Steven D. Sheetz, David E. Monarchi,Effectiveness of Traditional Software Metrics for Object-Oriented Systems, Department of Management Information Systems, College of Business Administration University of Denver, Denver, USA,2007.

[4] Jianguo Chen, Hui Wang, Yongxia Zhou, Stefan D. Bruda, Complexity Metrics for Component-based Software Systems, International Journal of Digital Content Technology and its Applications, vol.5, no.3, March 2011,pp.3-24.

[5] K. S. Jasmine, and R. Vasantha, A Quality Metric for Component based oftware Products, Journal of Systems and Software, vol.34, 2007.

[6] Kaur Chahal, K.Singh, Component Quality Model, Software Engineering,IEEE Transactions on Volume 20, Issue 6.

[7] Louis J. M. Taborda, The Release Matrix for Component-Based Software, Macquarie Graduate School of Management, Macquarie University Australia, 2010.

[8] Magnus Andersson Patrik Vestergren, Object Oriented Design Metrics, Journals of System and Software, vol.23, 1993.

[9] Ma Liangli, Wang Houxiang, Using Component Metadata based on Dependency Relationships Matrix to improve the Testability of Component-based Software, Department of Computer Engineering, Naval University, Wuhan,2009.

[10] Miguel Goulão, Fernando Brito e Abreu, Software Components Evaluation,Department of Information Technology, University Nova de Lisboa,Caparica, Portugal,2008.

[11] P. Edith Linda, V. Manju Bashini, S. Gomathi (2011), Metrics for Component Based Measurement Tools, International Journal of Scientific & Engineering Research Vol.2, Issue 5, May 2011.

[12] V. Lakshmi Narasimhan, P. T. Parthasarathy, and M. Das, Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE), International Journal of Informing Science and Information Technology, vol.6, 2009.