Volume: 7 **Issue:** II **Month of publication:** February

DOI: http://doi.org/10.22214/ijraset.2019.2138

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 7 Issue II, Feb 2019- Available at www.ijraset.com

Enhanced K-Classifier - A Framework to Measure the Reusability Metrics of Software

Saima Majeed
Department of Computer Science (M.TECH). Kurushetra University,Haryana,India.

Abstract: Reusability can be defined as a use of existing assets in some other form within the process of software product development. In other words, reusability of the component indicates the level to which the component can be reused in the other as well as same component based applications. On the basis of metrics which contribute towards the reusability factor, the training samples are prepared with the help of metric plug in. This research work is focused on building a framework which will help to measure the degree of reusability by using K means Classifier. The reusability of a component is calculated by using various samples and each of which falls in a category or a cluster like high to low reusability cluster. The proposed technique automatically calculates the reusability of the testing samples which are given as an input to the classifier with the help of enhanced K-means classifier.
Keywords: Evolution of Components, Component Integration, Role of Component in Reusability, Black box metrics

I. INTRODUCTION

Developing cost-effective and quality products is an essential and challenging piece of software product development .Component-based software development can help developers to efficiently produce software within the time and budget constraints. The theory of (CBSE) component-based software engineering is based on the development of independent and loosely coupled components of the system, by avoiding unrelated dependency among system components. [1].The component-based software development (CBSD) is accepted in the industry as a new effective software development paradigm. (CBSE) Component-based software engineering has been summarized by two development processes: the development of components for use and the development of component-based software systems (CBSS) with reuse by integrating components that have been deployed independently [2].Software component can be defined as a unit of composition in a narrow sense. Even without source code a user of a component in the form of an object code can exchange it independently. As the number of components accessible on the market increases, it is becoming more essential to devise software metrics to qualify the diverse characteristics of components. Software metrics are proposed to measure software quality characteristics quantitatively. Amongst several quality characteristics, the reusability is principally important when reusing components. It is required to measure the reusability of components in order to understand the reuse of components effectively [3].

A. Role of Component in Reusability

Components are the collections of many preprogrammed tools which are used as the add-on page which is to build use of those tools.

1) *Evolution of Components:* In 1990's, software systems for e-business transactions were built of components across numerous platforms, programming language and network protocols. No tools were available for detecting "bugs" or to determine the efficiencies in these programs (object oriented programs). Primarily the errors were printed using the output statement and the number of lines, number of packages and most important concepts of object oriented programs with the decisive programs Those programs which were used to measure the OOPS programs seem to be decisive and were implemented with coding. In 1997, Intermetrics, a software engineering company, proposed to build up a debugging tool for Component- based software systems whose main purpose was to identify the errors which would result in error free program..However it was not enough to give the therapy. Thus the problem to find the efficiency of the program is solved with the assist of the component relayed tools which efficiently provides the charts for the results and measures the program. The foremost advantage of component based tool is that the user can select their own tool to evaluate their program according to their needs. In this, the user can also see the links from where the tools are accessible, that is the user can also download or discern more about the tool from the direct link
2) *Component Integration:* All the tools won't rally the user required criteria. Some may be useful for generating report and some may be used just for evaluating the program. Some tool may evaluate the oldest method of measuring the program with some restricted parameters which are not in current use and which may be capable to measure. All the listed tools will measure the

common parameter such SLOC (Source Line Of Coding) and LOC (Lines Of Coding). It all depends on the programmer or the user who apply the component metrics. Integrating the diverse tools has many advantages and also some disadvantages. Those are listed below:

a) Advantages On Combining Components

1) Flexible
2) User friendly
3) Compatibility
4) Comparability

b) Disadvantages On Combining Components

1) Confusion may occur about which tool to be used
2) Some may not meet the user requisite
3) Some tools are only test versions
4) All the tools which are listed may have some common criteria which are not useful [4].

B. Metrics Used To Quantify Reusability

In quantifying reusability most authors relied on accessible well-known metrics. Some papers offer direct metrics for reusability which outcome in explicit quantitative or linguistic value depicting level of reusability. Some, however, offer indirect metrics, which give value for some other quality characteristic or measured factor, which again deeply influences reusability. Evaluating reusability is a matter of interpretation of given values in such cases.

1) Black box metrics: Black box metric component overall reusability (COR) combines metrics for adaptability, understandability and portability. Understandability is measured by (Existence of meta-information) EMI metric, adaptability by (Rate of Component Customizability) metric RCC and portability by SCCr (Self Completeness of Component's Return value).
2) White box / Glass box metrics: If we are owners of component, or we can attain component's source code some other way, we will have a large number of white/glass box metrics for measuring reusability at our disposal. When dealing with reusability coupling, cohesion and inheritance are considered. Another metrics based on fuzzy approach are coupling, volume, complexity, regularity, reuse frequency [5].

C. Benefits of Software Reusability

Although there are numerous benefits of software reusability, main benefits are listed below:

1) Improved quality
2) Minimized costs
3) Increased productivity
4) Reliability
5) Interoperability

D. Different Types of Reuse

1) Ad-hoc Reuse: When reuse occurs within projects, Ad-hoc reuse is preferred. It aims to save the time and reduce the redundancy.
2) Repository Based Reuse: When component repository is used which can be accessed by diverse application, Repository Based Reuse is done. This is based on quantity because any number of components can be put into the storage area and there is no control over their quality and usefulness. Exchange medium between the application groups is the repository [6].

II. RELATED WORK

V. Lakshmi Narasimhan et al[2009]Component-Based Software Engineering (CBSE) has shown significant prospects in rapid production of large software systems with enhanced quality, and emphasis on decomposition of the engineered systems with well defined interfaces which are used for communication across the components. In this paper, a series of metrics proposed by various researchers have been analyzed, evaluated and benchmarked using several large-scale publicly available software systems. A systematic analysis of the values for various metrics has been carried out and several key inferences have been drawn from them. A number of useful conclusions have been drawn from various metrics evaluations, which include inferences on complexity, reusability, testability, modularity and stability of the underlying components. The inferences are argued to be beneficial for CBSE-based software development, integration and maintenance.

MajdiAbdellatief et al. [2013] a component-based software system (CBSS) is a software system that is developed by combining components that have been deployed independently. Although there are various metrics proposed by Researchers during last few years to evaluate CBSS attributes. However, the practical use of these metrics can be complex. For example, some of the metrics have concepts that both overlap or are not well defined, which could hinder their implementation. The main aim of this study is to understand, classify and examine existing research in component-based metrics, where the focus should be on approaches and elements that are used to estimate the quality of CBSS and its components from a component consumer's point of view. This paper presents an organized mapping study of several metrics that were proposed to measure the quality of CBSS and its components. Author found 17 proposals that could be applied to estimate CBSSs, while 14 proposals could be applied to evaluate individual components in isolation. Only a few of the projected metrics are soundly defined. The quality assessment of the primary studies detected many restrictions and suggested guidelines for possibilities for improving and increasing the recognition of metrics. Evaluating and characterizing CBSS and its components quantitatively still remains a challenge. Thus much effort can be made to attain a better evaluation approach in the future.

Hironori Washizaki et al.[2003]in component-based software development, it is necessary to measure the reusability of components in order to realize the reuse of components efficiently. For measuring the reusability of Object- Oriented software, there are some product metrics. However, in application enlargement with reuse, it is difficult to use conventional metrics because the source codes of components cannot be produced, and these metrics require analysis of source codes. In this paper, author propose a metrics suite for measuring the reusability of such black-box components based on limited information that can be obtained from the outside of components lacking any source codes. Author defines five metrics for measuring a component's adaptability, understandability and portability, with confidence intervals that were set by statistical analysis of a number of JavaBeans components. Moreover, author provides reusability metric by combining these metrics based on a reusability model. As a result of estimation experiments, it is found that our metrics can effectively recognize black-box components with high reusability.

P. Edith Linda et al.[2011] the main aim of this paper is to integrate the different object oriented metric tools and make them accessible as a single add-on. The first part of this paper analyzed five dissimilar tools and they are migrated into one to make use of those tools in efficient manner.

Marko Mijac et al.[2015] Reusing software assets have lots of advantages and has been essential feature of all software development approaches. Component based software development has been principally inspired by reuse. In order to reuse software component, the component has to be intended and built for reusability. Since reusability is influenced by a number of different factors, there are different approaches and metrics used to measure reusability. In this paper author conducted extensive literature review in order to recognize reusability metrics and factors influencing reusability. sum of 39 papers introducing reusability metrics were originate and analyzed. Author identified 36 different factors influencing reusability, more than 20 white box/glass box metrics and 12 black box component metrics.

Swati Thakral et. al [2014] The paper demonstrates a literature review of diverse software reusability concepts. It provides a brief and systematic review of available reusability metric projected by different researchers in different journals and conferences. The Objective is to gather useful information on software component reusability and the factors rely on which reusability of the component is highly dependent. As a outcome of literature review we create that reusability is highly dependent on customizability, interface complication, documentation quality and portability.

V. PrasannaVenkatesan et al.[2009] Software metrics can provide an automated way for software practitioners to evaluate the quality of their software. The earlier in the software development lifecycle this information is presented the more valuable it is, as changes are much more expensive to make later on in the lifecycle. As far as the Component-Based Software Engineering is based, the metrics can help estimate, plan and identify areas to develop quality, reduce costs, enhance project management and facilitate risk management. Ultimately the successes of the CBSE projects can be evaluate from the metrics. In this paper author define seventeen metrics for seven component characters. It consists three functional characters namely the suitability, accuracy and complexity and four non-functional characters specifically the usability, maintainability, reusability and performance. The metrics are arrived at, relayed on a metric model. The metrics are then tested with a case study.

William B. Frakes et al.[2005] This paper briefly summarizes software reuse research, discusses main research contributions and unsolved problems, provides pointers to main publications, and introduces four papers chosen from The Eighth International Conference on Software Reuse (ICSR8).

Adnan Khan et al. [2014] Component-based development allows us to develop and integrate product components which facilitate software reusability, elevated quality and generalization for testing. Component-Based Software Engineering makes use of approaches which are relayed on architecture definition languages, object oriented propose and software architecture. These strategies support in the development of both domain-specific and generic software products. Reusability approach speeds up software development by using previously developed components, thus software development cost and time is substantially reduced. In this paper, author presents a Component-Based Software Engineering framework for software reusability.
AnshulKalia et al.[2014]the reusable software components can be defined in some ways. The reusable software components possess a different functionality that does not affect the functionality of other components. It has also been specified precisely that for what the component recycle stands for and for what the component reuse does not stands for. It is essential to characterize the components for improved reuse. Characterization describes the features and characteristics of components. Distinct components show distinct characteristics in diverse domains of their usage and in different operating environments. The components can be classified on the basis of properties it have, that facilitates with the better usage, improved retrieval, enhanced understanding and better cataloguing. Through component classification one gets the assurance of selecting right component and it suggests diverse ways in which a component can be reused. The paper explains for the requirement of characterization which ultimately is reflected from the above stated fact. Besides that it gives the criteria to characterize the reusable components. The standard is laid down while keeping in mind the general specifications and formal specifications. These stipulations in one way determine the feasibility of a reusable component at the initial level. The common and formal specifications depict the internal and external characteristics which play a significant task in the identification, selection, adoption and implementation of components in a particular application development. Specifications will tell the true nature of a reusable component that helps to outline the required components. It also discusses the pressure of characterization that it puts on the reuse of reusable components.
Marcus Kessel et.al[2015] Pragmatic software reuse, in which accessible software components are invasively adapted for use in new projects, involves three major activities – selection, adaptation and integration. Most of the academic research into pragmatic research to date has stressed on the second of these activities, adaptation, especially the definition of reuse plans and verification of invasive changes, even though the collection activity is arguably the most important and effort-intensive of the three activities. There is therefore a grand deal of scope for improving the level of support provided by software search engines and recommendation tools to pragmatic re users of software components. Test-driven search engines are particularly promising in this regard since they acquire the inherent ability to “evaluate” components from the perspective of users’ reuse scenarios. In this paper author discuss some of the main issues involved in improving the selection hold up for pragmatic reuse provided by test-driven search engines, describe some new metrics that can help address these issues, and present the outline of an strategy for ranking software components in search results.
A.Aloysius et al. [2015] in the technological world every day number of software is developing and available in the market but measuring the reusability of the software is still a very big challenge. Component based software is emerging field and now-a- days, mainly of the software are developed by using the technique of component based software development (CBSD). So that the difficulty, time, error factors were reduced and reusability is achieved. The success of the CBSD projects can be confirmed only from the metrics that are previously projected. In this paper, various component based metrics projected by the researchers have been discussed and then suggested the future enhancement.

III. PROBLEM FORMULATION AND OBJECTIVES

To prepare a complete software project, various components are required. Most of the components are developed in such a way so that they can reused and can be used in any another project if needed and if one component is not functioning properly it can be interchanged with other component by plugging in and out with other components which will make maintenance easier. There is an urgent need to understand how these software components can be implemented as plug and play devices and reusability of software components can be understand in some tangible framework. Therefore, in this research work we are solving this issue by building a framework which helps to measure degree of reusability by using clustering methods (machine learning) , which would be best suited for our problem definition and better from previous research works.

A. Objectives

1) Identify parameters which impact the reusability of software components.
2) Using machine learning algorithm to develop automated assessment of reusability.

IV. METHODOLOGY

A. Selecting Projects For Developing Representative Datasets

For any software analysis the basic step is to create the database. At this step we will develop the representative datasets so that we could evaluate metrics and then reusability could be measured.

B. Downloading The Projects From Git Repository

In this step we are getting open source GIT local repository and the projects which are stored in Eclipse.

C. Github Open Source Project Repository

Git stores and thinks about information much differently than these other systems like the major difference between Git and any other repository Subversion and friends included is the way Git thinks about its data. File based systems store the information as a list of changes. These systems CVS, Subversion, Perforce, Bazaar, and so on think of the information they remain as a set of files and the changes made to each file over time, as illustrated in Figure 4.1

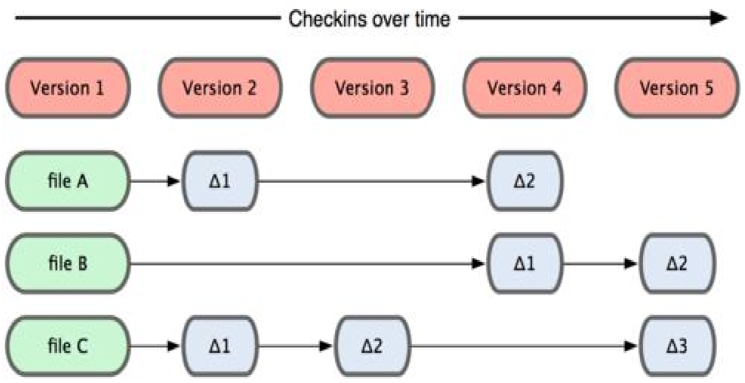

Figure 4.1

Git doesn’t assume or store its data this way but stores like a set of snapshots of a mini file system. Every time you commit, or save the state of your project in Git, it chiefly takes a picture of what all your files look like at that moment and stores a reference to that snapshot. If the files have not changed, Git doesn’t store the file again in order to be more efficient—just a link to the previous identical file it has already stored.

D. These Are The Following Total Metrics We Have Selected

Name of metric Reusability	Relative value of metric	Implication
LCOM	increases	increases
WMC	increases	decreases
NOC	increases	increases
DIT	increases	increases
CPD	increases	decreases

Table 4 Showing The Selected Metrics

Inferences from the LCOM metrics (Lack of Cohesion in Methods): For high value of LCOM metric, reusability of that component is high and the component is relatively less complex.
Inferences from the WMC metric (Weighted Method for Class): For High value of WMC, reusability is considered low.
Inferences from the NOC metric (Number of Children): A high value for the NOC metric implies that the components are highly reusable.
Inferences from the DIT metric (Depth of Inheritance Tree): If the value of DIT is high, reusability is high and complexity is high.
Inferences from the CPD metric (Component Packing Density): For high value of CPD metric implies that the reusability decreases.

V. RESULTS AND ANALYSIS

Table 5.1 Dataset Used For Training (K-Means)

Cluster	LCOM	WMC	NOC	DIT	Reusability
1	0.303	34	1	2	Low reusable
2	0.265	6.511	47	2.383	Highly reusable
3	0.599	11.474	38	4.211	Highly Reusable & Less Complex Component
4	0	2.667	9	1.222	Low Reusable & Highly Complex Component
5	0.333	18	2	7	Highly Reusable & Highly Complex Component

TABLE 5.2 TEST SAMPLES

LCOM	WMC	NOC	DIT
0.467	13.455	22	3.864
0.14	8.703	12.333	2.324
0.482	13.75	20	2.85
0.41	5.143	14	2.714
0.353	10	4	1

Table 5.3 Results Of K-Means Classifier

MEMBER OF CLUSTER	LCOM	WMC	NOC	DIT	REUSABILITY
3	0.467	13.455	22	3.864	Highly Reusable & Less Complex Component
4	0.14	8.703	12.333	2.324	Low Reusable & Highly Complex Component
4	0.482	13.75	20	2.85	Low Reusable & Highly Complex Component
4	0.41	5.143	14	2.714	Low Reusable & Highly Complex Component
4	0.353	10	4	1	Low Reusable & Highly Complex Component

At this step we have downloaded 10 projects from internet which will help in developing the representative datasets so that we could evaluate metrics and their reusability could be measured. The following 10 projects out of which first 5 are for training and another 5 for testing. Using Eclipse, we run each of the below java projects and collect the various metric values like WMC, NOC, DIT LCOM etc. When we run the algorithm, we find the expected values called predicted values corresponding to our actual values for each observation of every project. K-means algorithm- K-means is one of the simplest unsupervised learning algorithms which solve the clustering problem. The procedure follows a simple and easy way to classify a given data set in a certain number of clusters (assume k clusters). For each cluster, we can define k centroid which is the main motive. We can place the centroids far way from each other as much as possible. The next step is to take each point from a given data set and associate or link it to the nearest centroid. When all the points are considered, then the early grouping is done. At this point we re-calculate k new centroids of the clusters formed by the previous step. After these k new centroids, a new binding is done between the same data set points as well as the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step or again n again until no more changes are done. After the formation of above 8 clusters, we give the next 8 projects metric values to our classifier and after performing all its calculation i.e. comparing the metric values of every project with every other cluster, it assigns each of the projects to that cluster which has similar characteristics as that of the project.

VI. CONCLUSION AND FUTURE SCOPE

In this research, we have proposed the technique which automatically calculates the reusability of the components or projects with the help of enhanced K-means classifier. On the basis of metrics which contribute towards the reusability factor, the training samples from 5 clusters and when the testing samples go through or given as input to the classifier, they all lies in different cluster according to their matching features with their corresponding cluster as shown in table5.3.In this paper ,the proposed approach calculates the reusability of the components (samples) and each of the sample falls in a category or a cluster like high to low reusability cluster but the samples were taken in this research are small. So in future, this approach or technique can be enhanced or extended for better performance of evaluation for large databases or samples.

REFERENCES

[1] V. Lakshmi Narasimhan, P. T. Parthasarathy, and M. Das, “Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE)”, Issues in Informing Science and Information Technology Volume 6, 2009
[2] MajdiAbdellatief, Abu BakarMd Sultan, Abdul Azim Abdul Ghani1, Marzanah A. Jabar,”A mapping study to investigate component-based software system metrics”,The Journal of Systems and Software,2013, pp. 587-603.
[3] Hironori Washizaki, Hirokazu Yamamoto and Yoshiaki Fukazawa,”A Metrics Suite for Measuring Reusability of Software Components”,Ninth International Software Metrics Symposium,IEEE,2003, pp.1-13.
[4] P. Edith Linda, V. ManjuBashini, S. Gomathi,”Metrics for Component Based Measurement Tools”,International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011.
[5] Marko Mijac,ZlatkoStapic,”Reusability Metrics of Software Components: Survey”,Research gate publications,September 23-25, 2015,pp.221-231
[6] Swati Thakral, ShraddhaSagar and Vinay, ”Reusability in Component Based Software Development - A Review”,World Applied Sciences Journal 31 (12): 2068-2072, 2014, pp.2068-2072.
[7] V. PrasannaVenkatesan, M. Krishnamoorthy,”A Metrics Suite for Measuring Software Components”,Journal of Convergence Information Technology Volume 4, Number 2, June 2009,pp.1-16
[8] William B. Frakes and Kyo Kang,”Software Reuse Research: Status and Future”,IEEE Transactions on Software Engineering, Vol. 31, No. 7, July 2005,pp.529-536
[9] Adnan Khan, Khalid Khan, Muhammad Amir and M. N. A. Khan,”A Component-Based Framework for Software Reusability”,International Journal of Software Engineering and Its Applications Vol. 8, No. 10 (2014), pp. 13-24.
[10] AnshulKalia, SumeshSood,”Characterization of Reusable Software Components for Better Reuse”,IJRET: International Journal of Research in Engineering and Technology,Volume: 03 Issue: 05, May-2014, pp.584-588.
[11] Marcus Kessel, Colin Atkinson,”Ranking Software Components for Pragmatic Reuse”,2015-16th International Workshop on Emerging Trends in Software Metrics ,IEEE,pp.63-66
[12] A.Aloysius and K.Maheswaran,”A Review on Component Based Software Metrics”, Intern. J. Fuzzy Mathematical Archive Vol. 7, No. 2, 2015, pp.185-194