APTIKOM Journal on Computer Science and Information Technologies
Vol. 1, No. 2, 2016, pp. 63~76
ISSN: 2528-2417
DOI: 10.11591/APTIKOM.J.CSIT.16

Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design

Manoj H. M.[1]*, Nandakumar A. N.[2]
[1]Dept of CS&E, Don Bosco Institute of Technology, Bangalore, India
[2]R. L. Jalappa Institute of Technology, Dhoddaballapur, Bangalore, India
*Corresponding author, email: manoj.hmhm@gmail.com

Abstract

The role of design pattern in the form of software metric and internal code architecture for object-oriented design plays a critical role in software en-gineering in terms of production cost efficiency. This paper discusses about code reusability that is a frequently exercised cost saving methodology in IT produc-tion. After reviewing existing literatures towards study on software metrics, we found that very few studies are witnessed to incline towards code reusability. Hence, we developed a simple analytical model that establishes relationship between the design components of standard software metric and code reusability using case studies of three software projects (Customer Relationship Management project, Supply Chain Management project, and Enterprise Relationship Management project). We also testify our proposal using stochastic based Markov model to find that proposed system can extract significant information of maximized values of code reusability with increasing level of uncertainties of software project methodologies.

***Keywords**: Analytical Modelling, Code Resusability, Design Pattern, Software Methodologies*

1. Introduction

In today's world, every sector of industry or services is dependent on the computer-based applications. In order to improve performance and gain competitive edge, quality of the software has become a crucial factor. Developing and outsourcing of software service is a major and rapidly growing industry in many part of the world [1]. The process of software development is described through the term software engineering that refers to usage of a systematic procedure in context to standard set of goals for performing analysis, designing, implementation, and testing as well as maintenance of the software. The software thus developed must be reliable, useable, efficient, modifiable, testable, maintainable, interoperable, portable and accurate [2]. In the process of software development, object oriented design is considered as one of the important feature to evaluate quality of software [3]. Irrespective to size of the organization, object oriented design methodology is majorly adopted for software development in any organization. Therefore, object oriented designs are considered as standard through which system objects can have particular features and also necessary characteristic. The reason behind adoption of object oriented methodologies is that it allows to visualize the problem and acquire a solution in all macro and micro level in related to objects and also ensuring better reliability, adaptability, flexibility and reusability [4]. Currently software engineers use software metrics to evaluate design component and necessary resources of a certain software project. The advantage of software metric is that it allows the evaluation of design pattern through better platform as well as assistance in performing the testing of application in quantitative manner. Through such testing, the reliability of the software can be demonstrated. In general, when a company receives a new requirement from client, initially they will formulate a design of the requirement. This confirmed design from the architect is sent for production. On completion of coding, the product is dispatched to the customers. Although it is unethical to reuse the code of earlier client in order to develop application for new client [5], but code reusability also deals with security of intellectual property. A production team has to go for a fresh development starting from the scratch, which not only requires effort but also considerable amount of time and money. Majority of large organization now use design pattern which are subjected to reuse without the ethical issues. The main objective of the design reuse is to assist the developer to use it in the new production, which helps in cutting down the new

Received March 15, 2016; Revised April 3, 2016; Accepted May 8, 2016

development cost starting from scratch. However when reusing the design care should be taken so that the design is optimally reused for the current as well as the future client. Adopting design reuse will also help in ensuring the timely production and delivery process. Design reuse does not imply that the entire design is used; it might be like some percentage of the current design is used in the new or future similar project. Hence the focus of the design team should be such that ,the design is not only focused on existing client but it should be able in providing a minimum proportion of reusable design for future clients also. But in reality it is not easy since the future requirement of the client is unpredictable.

In section 2 gives an overview about the software metrics. Section 3 highlights about the CK Metric key points. Section 4 discusses about the related work. Followed by Problem identification in Section 5. Section 6 discusses about proposed system. Section 7 discusses about the research methodology. Section 8 provides the outcome and result analysis of the proposed system and Finally Section 9 provides conclusion and future work.

2. About Software Metrics

In this era of IT revolution, software development as emerged has crucial requirement in every phase of day to day life. From academics to public service, healthcare to banking, entertainment to sports, the use of software is involved in one or other way contributing to the advancement of the domain. Various factors that have contributed in the advancement of software development are its key features like flexibility, portability, design reusability, software metrics and so on. Design reusability plays a vital role in software development, since it is not only involved in the development of current work but also lays a blueprint for future requirement of the current as well as new projects. Design reusability will also help in reducing the cost of production and reducing required manpower and development time by providing a framework that consist of the design which can be reused over a period of time. Another significant factor of software development is the software metric, which defines standard degree of measure to which the process or software system will possess certain property. In software engineering different metrics are available to measure different parameters such as process is measured using process measurement, project using project measurement and product metric to measure product metrics.

In our work, since object –oriented deign is involved the paper will highlight few things related to Object-oriented metrics. To develop metric for the object-oriented design, seven different measurable qualities are listed below [6].

a. Complexity: Analyzed by assessing the way classes are related to each other.
b. Coupling: It is the physical connection between the object oriented elements.
c. Sufficiency: Its defines the degree to which the abstraction should possess the features needed by it.
d. Cohesion: It is determined by analyzing the group of properties posses by the class being the part of problem domain or design domain.
e. Primitiveness: Used to indicate the degree of atomic level of operation.
f. Similarity: It is used to identify the similarity between the classes in the term of behavior, structure, purpose or function.
g. Volatility: used to define the probability of happening of change in object oriented design.

2.1. Size

Using four different perspectives such as volume, length, population and functionality. The measurement of population is done by evaluating the total number of OO entities, which is in the form of classes or operations. Measurement of volume is achieved dynamically at any instance of time. Functionality denotes the value provided to user by object oriented application. Using interconnected design like depth of inheritance tree length is measured. Object oriented design metrics concentrates on measurement that is applicable to class and design characteristic. Through these measurements designers are permitted to access software in early process phase, allowing making changes that minimizes complexity and enhances the continuing ability of design [7].

3. CK-metric

In 1994 Chidamber and Kemerer introduced standard software metric for object oriented programs. CK metrics plays a vital role in knowing the design aspects of software and improving the software quality. The main objective of the CK metric is to provide an in detail measurement of cumulative quality of the software programs to every class level. Metric is associated with each and every tiny segment of the software providing the overall information of the software quality. In CK metric, six classed based metric for object oriented programs as follows:

a. Weighted Methods per Class (WMC)
   Used to define, sum of complexity in a class. In whole it represents the complexity of the class and this measure can be utilized for indicating development and maintenance effort for class.
b. Reponses for a Class (RFC)
   This metric represents number of growing methods within a set, which can be called in response to message sent to an object performing certain task.
c. Depth of Inheritance Tree (DIT)
   This is one of the frequently used metrics; it is used to estimate the extent of depth in the hierarchy of class. It is also used in evaluating maintainability and reusability.
d. Number of Children (NOC)
   This is a measure of number classes that are associated with a particular class with assistance of inheritance relationship. A class with many children implies a bad class with bad design.
e. Coupling between Object Classes (CBO)
   This defines the number of all other set of classes for which it is coupled. CBO is advantageous in determining the complexity in testing and reusability.
f. Lack of Cohesion of Methods (LCOM)
   It is the difference between the number of methods in which the similarity is zero and the number of methods in which the similarity is non zero. Similarity between the two methods is the number of feature that is being used in common. A zero in LCOM does not signify cohesiveness as well a high value does not represent any inference. In LCOM it is difficult to define a unit and to measure quality. LCOM is not recommended for accurate measurement since it does not quantify quality properly.

Object oriented metrics are being successfully used in different domains and programming languages in different parts of the world. These metrics have consistently illustrated the relationship to quality factors like reuse, defects, cost as well as maintainability and relationship which may go beyond the size. The evaluation of these metric are achieved through certain tools like metamill [8], metric 1.3.6 [9], Analyst 4J [10], OOmeter [11] and Dependency finder.

4. Review of Literature

This section discusses about the prior literatures where various approaches of software metric design and its contribution has been introduced. Many approaches have been developed over the years to address the problem of detecting and correcting design flaws in an Object Oriented (OO) software system using metrics. Moreover, with the ever increasing number of software metrics being introduced, the project managers find it hard to interpret and understand the metric scores. As Object Oriented Metrics require very good understanding of Object-Oriented concepts and moreover, there is no single metric present which gives all features of Object-Oriented Software System. The Table 1 shows the existing survey on design flaws in an object oriented (OO) software system using metrics. Table 1 will highlight the tabulated discussion of various problems and respective techniques used to enhance the performance of software metrics in software engineering.

Table 1 Existing Studies towards Software Metrics

Authors	Problem Focused	Techniques used	Performance parameters
Basili et al. [12]	examined the suite of Object-Oriented proposal metrics presented by Chidamber & Kemerer.	Empirical validation	C&K OO metrics gives better predictor than conventional metrics.
Anna et al. [13]	to measure the reusability and maintainability degrees of aspect-oriented systems.	Empirical and quantitative analysis, Aspect-oriented software development (AOSD).	Degrees of complexity, diverse domains.
Kaur et al. [14]	To exploring structural factors for software components.	Software metrics using neural networks.	exhibit an efficient model targeted for software programmers.
Kaur et al. [15]	classification and assessment of various reusability metrics.	K-Nearest Neighbor-based technique.	
Kumari et al. [16]	To compare C++ and Java programs.	Statistical inference techniques, Object-oriented metrics.	More realiability.
Linda et al. [17]	To analyze the different metric suites for object oriented schems.	Development of a software prototype like "Class Break point Analyzer (CBA)"	Software quality, realiability.
Wu Et al. [18]	To do comparative analysis on on C++, C#, and Java programs by using object-oriented Metrics.	Comparative study on software metric reusability in software engineering.	Reusability and Realiability.
Srinivasan et al. [19]	To analyze and reviews the most referred object-oriented metrics in software measurement.	Done Comprehensive Review on object oriented metrics using for software measurement.	
Scotto et al.[20]	To evaluate the effectiveness of the metrics.	Web Metrics for SQL queries.	
Singh et al.[21]	To estimate the reuse of software matric.	LMT (Logistic Model Trees).	reusability
Subramanyam et al. [22]	To reduce the computational complexities in object-oriented programs for identifying defects	object-oriented programming	Good enhancement, complexities.
Shaik et al. [23]	To itemize and maintenance of software.	Object Oriented Software Systems	The effort, different metrics
Zahara et al. [24]	To examine the competence and effectiveness of machine learning regression techniques.	Multi-linear regression, "Standard instance-based learning IBk with no distance weighting"	Software crisis, software quality, productiviey.
Manoj et al. [25]	To contemplate about software metrics.	An extensive literature survey on ranking code reusability in software engineering.	Cost effecitve, quality, design complexity.
Oberoi et al.[26]	Analyzing CK metric values of component-based software	Software component design patterns, Self-Organizing Map and empirical evaluation.	Optimized metric values.
Goyal Et al. [27]	To find out the reusability value for software.	Unsupervised neural network.	Reuability, complexity,
Jayalakshmi Et al. [28]	To measure quality in terms of software performance and reliability.	Functional parameters and non-functional parameters.	complexity, reliability and robustness of the software
Hudly Et al. [29]	To evaluate the main features of object oriented like Polymorphism, Encapsulations, Data abstraction, Inheritance and classes.	Two kinds of metrics: classes and to measure the class design configuration of the program.	complexity, reliability
Liu Et al. [30]	The gap between quality measurement and design of these metrics.	Object Oriented Designs software.	Safety, complexity.
Paliwal Et al. [31]	To increase the productivity and maintainability of any software.	Reusable module.	Module Reusability, Dependencies, Class size.
Chauhan Et al.[32]	to assess software quality at design level.	14 Java Parser, Eclipse with Metrics 1.3.6	quality of software.
Gupta et al. [33]	a comparative study of many software quality estimation methods.	Fuzzy Logic techniques, artificial neural network (ANN),	Better quality of software
Alcalá et al. [34]	Improving the performance and flexibilyzing the model structure.	Linguistic model structure using weighted double-consequent fuzzy rules.	complexity, reliability

5. Problem Identification

Software engineering has played a significant role in successfully delivering the quality-oriented project. The existing literatures discussed in prior section discusses about various techniques for enhancing the crude performance of software metrics. It is found that majority of the techniques uses quantitative, empirical, tree-based techniques. Some of the unique evolutionary techniques e.g. neural network, fuzzy logic etc has also been used. All these above techniques have possible advantages as well as disadvantages too. This section we will discuss about the problems being identified after reviewing the existing system as follows,

5.1. Lack of Benchmarked Research

Except CK-metrics introduced during 90s, we have not come across any robust discussion of software metrics especially for object oriented programming. Although there are presence of massive volumes of papers in internet media, but few authors have been found to introduce any novelty in their ideas. Some of the idea was to implement CK metric or introduce a new mathematical formula over the old equations of parameters e.g. DIT, WMC, NOC etc. Also, we have not come across any research model which was found to use or followed by other researchers much towards code reusability.

5.2. No studies towards CK Metric Relationship

100% of all the papers introduced towards software metrics suites have their own formulations. The authors normally check for problems associated with CK metrics and attempt to introduce new software metric suite. However, there was no significant attempt in the past to investigate the relationship among the CK metrics towards code reusability, which is very common operation in any IT industry or any production company.

5.3. Lack of focus on Code Reusability

Code reusability is the common practice in any IT production. However, it has received very less attention among the research communities worldwide. Studies towards code reusability with respect to its possible relationship with the software metrics are less explored topic. These above problems are addressed in this paper in the form of a simple formula with an aid of case study. The next section discusses about the proposed system that enhances the performance of the code reusability and enables the user to visualize enough statistics between software metrics and code reusability.

6. Proposed System

The proposed study aims to develop an analytical framework which establishes a relationship in between different type of CK-Metrics components with code reusability. The proposed system includes two different type of experimental prototyping which are i) modelling for the estimation of code reusability and ii) frame work to evaluate the impact of design metrics on code reusability. The proposed study has been highly motivated by all the studies that have been carried out in past which represents that the improvement in the software quality can be achieved through performing efficient code reusability and implementation of various measurement driven modules. The modelling for estimation of code reusability index considers a flow of processes where CK metric data from design and code artifacts and domain knowledge and experience are further processed through empirical analysis. The CK-Metrics data can be acquired from UML (Unified Modeling Language) class diagrams or the equivalent java codes. There various processing tools like Rational Rose etc, which are used to extract the metric data from code artifact's and anticipated to bridge a relationship in between the CK-Metrics components such as WMC (Weighted Methods per class), DIT (Depth of inheritance Tree), NOC (Number of Children), CBO (Coupling Between Object Classes), RFC (Response Set for Class), LCOM (Lack of Cohesion in methods) etc. The empirical analysis generates data for the influence of property on code reusability and establishes a relation between various multifunctional estimation equations belong to different type of CK-Metrics components and maps them with code reusability. Finally a software framework for estimation of code reusability has been implemented using GQM paradigm and weighted factors from design metrics. The code reusability model has been further processed through a framework which calculates different type of CK-Metrics components using static class diagram and dynamic sequential diagrams. The proposed model also evaluates a data related empirical analysis and the analysis further generates the influence of individual metrics on code reusability, the linear combination of coefficients with respect to individual design metrics. The proposed system also takes CK-Metrics data as input and evaluates the code reusability index which can be further mapped into CK-Metrics components during a software development lifecycle. There are various existing conventional studies which are based on code reusability design metrics using empirical prototyping. The relationship between various CK-Metric components and the code reusability has been developed based on framing following conventional myth as follows,

a. It can be seen that classes with higher Depth of Inheritance Tree values will have higher probability of code reusability.
b. Classes with low cohesion results better software design and code reusability.
c. A class which consists of higher values of WMC and NOC extends the code reusability in dynamic scenario of software development.
d. Classes with higher CBO and RFC values increase the computational complexity and maximize the code reusability.

The proposed model also introduces a framework for code reusability which has been evaluated using deep empirical analysis and data modelling. The Empirical model considered two different types of medium high level projects where an experimental analysis has been carried out considering a huge number of classes to investigate the code reusability of the designed metrics. The classes associated with each projects configured and grouped in terms of different metrics values to avoid the intellectual property issues. The proposed study developed an effective and computationally efficient framework. The contribution of the proposed study includes i) ensuring the estimation of code reusability on heterogeneous object oriented software modules, ii) calculating the linear combination of weighted polynomial equations, iii) formulating an efficient relationship in between the CK-Metrics components and the code reusability. The performance metrics associated with the empirical model has been evaluated which ensures the effectiveness of the proposed system. The next section will discuss about the research methodology which formulate the relationship between design quality metrics and code reusability in detail.

7. Research Methodology

The proposed system is designed with an aid of analytical modelling approach as a standard of research methodology. The design of the proposed system is based on code reusability concept using CK-metric suite. Figure 1 highlights the modelling of the code reusability where the extraction of CK metric data is done from UML. A multiple forms of java-related cases can be used for obtaining CK metric. Various industry-related tools like Metamill, Metric 1.3.6, rational rose etc can be utilized in this regards to get the components of CK-metrics. However, class diagram can be manually used for estimating the number of classes. There are various parameters that can be evaluated with an aid of static classes e.g. DIT, WMC, and NOC, whereas various forms of sequential diagram can be used for evaluation other metrics e.g. RFC and CBO. However, it is quite imperative that LCOM couldn't be assessed or evaluated from the design patterns e.g. UML directly. However, sophisticated industry-based automated tools can be used for the same reason.

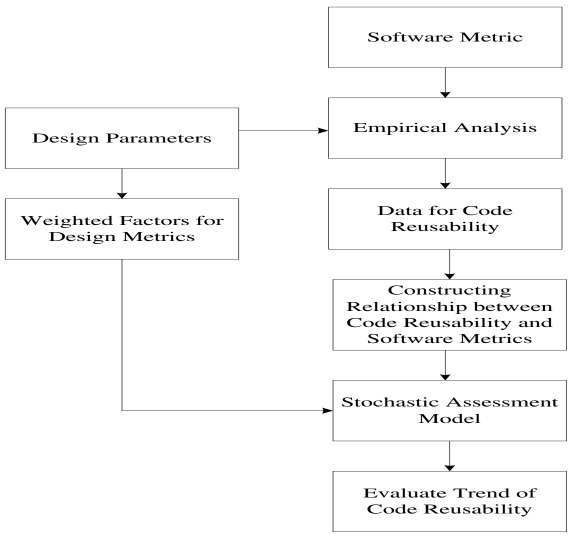
Software Metric
Design Parameters
Empirical Analysis
Weighted Factors for Design Metrics
Data for Code Reusability
Constructing Relationship between Code Reusability and Software Metrics
Stochastic Assessment Model
Evaluate Trend of Code Reusability

Figure 1. Schematic Diagram of Code Reusability

In order to assess such design-related issues i.e. code reusability, we consider sample projects developed in Java with significant number of classes. The proposed system targets to understand how individual components of CK-metrics affect code reusability. We develop a simple function in order to establish a relationship between metrics and code reusability. We use the concept of weighted coefficient as well as linear approach for assessing the possible impact analysis of CK-metrics over code reusability. We also use the concept of GQM (Goal-Question-Metric) as the core part of the research methodology that allows formulating the conceptual level of code reusability based on operational level and quantitative level.

7.1. Hypothesis Design

As discussed earlier that proposed study intends to understand the underlying relationship between components of CK metrics with code reusability, hence, an appropriate hypothesis is constructed for this purpose. The study performs analytical assessment on various ERP (Enterprise Resource Planning) and SCM (Supply Chain Management) related software projects on Java and following null hypothesis is being constructed.

a. H_{o1}: Better code reusability can be retained by moderate value of DIT in every class.
b. H_{o2}: The complexity in code design and reusability decreases for maximized values of RFC.
c. H_{o3}: Code reusability degrades for increasing values of NOC present in class.
d. H_{o4}: The hypothesis is increased CBO values doesn't have much impact over code reusability
e. H_{o5}: Code reusability declines for increasing values of WMC in every class.

7.2. Developing Analytical Modelling

The development of the proposed system is carried out using analytical modelling approach for testing the hypothesis. The system also applies simple mathematical estimation techniques to investigate the possible relationship between the components of CK metrics and code reusability. The proposed system considers case study of ERP and SCM related software projects developed on Java. The total number classes considered for ERP software project is 220. There are around 180 bases classes in it. Similarly, there are around 570 total classes and approximately 380 maintainable classes for SCM software project. Although, our technique could include more number of software projects, but we choose to consider using only ERP and SCM software projects. The components of the CK metric have multiple values which can be arranged or structured more appropriately. The proposed system performs simple analytical modelling for code reusability in the form of,

$$C_r = \frac{\beta \times 100}{\alpha} \tag{1}$$

The above equation (1) represents mathematical representation of code-reusability, where α is considered as total amount of classes available in every CK metric, whereas β is the component of CK-metric that corresponds to amount of classes newly designed by incorporating code reusability. This will mean that higher the value of C_r, higher is the extent of code-reusability. The evaluation of the α and β parameter is carried out manual as well as recording the same over spreadsheet. However, the values considered for discussion in result analysis is approximated to get contrastive outcome for investigating the effect of components of CK metrics over code reusability in software engineering. Finally, the analytical modelling is also testified with respect to presence of uncertainty.

8. Result Analysis

The analysis of the proposed study was carried out over SPSS [35] tool. We perform both numerical analysis as well as graphical analysis to assess the effectiveness of the outcome. Table 1 shows the numerical outcomes of the considered case study of ERP and SCM software projects, where the necessary CK metric components were closely observed and computed for code reliability using the simple equation 1 illustrated in prior section. Following are the discussion of the graphical outcomes of proposed system.

8.1. Effect of DIT on Code Reusability

The outcome shown in Figure 2 highlights that there is a significant improvement of code reusability for the DIT values. A closer look into the numerical values of DIT shows that with increasing trends in value of parameters α and β, the code reusability enhances significantly. The increasing number of levels of DIT will definitely represent little bit of complexity in computation; however, it is definitely productive to make the code more reusable. It is also suggested that for massive objective oriented design, it is quite possible that number of classes drastically increases, which also increases the possibility of DIT values. However, using the proposed system, using moderate values of DIT can actually retain better trends in code reusability. Therefore, the null hypothesis stating that "*Better code reusability can be retained by moderate value of DIT in every class" is found to be accepted and true*".

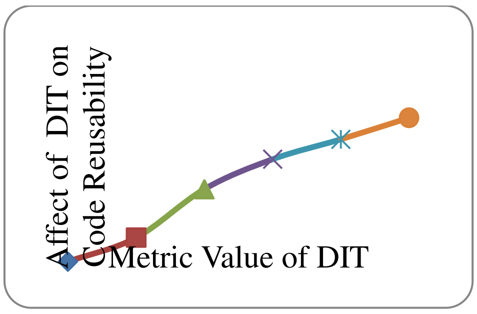

Figure 2. Affect of DIT on Code Reusability

8.2. Effect of RFC on Code Reusability

Figure 3 shows some interesting trends of code reusability. Although, the trend of C_r is found to be increasing but the trend is not smooth enough in the preliminary values of RFC metric. The basic trends explored here is that with increase of α and β parameters has witnessed enhanced code reusability factor. However, it also brings complexity in the major ranges of metric values (1-7), which will mean that although code reusability increases but it also brings significant complexity over design. For our analysis purpose, we use two forms of data to generate the graphical outcomes. The first data is synthetic data and can be seen in the RFC row in Table 1, where the parameters α and β are maintained in combination of both increasing and decreasing order. The outcome shows increase of code reusability, however, it is less likely that for complex ERP and SCM projects, the values of α and β are usually in increasing order. Hence, we plot a graph by using the increasing order to α and β to investigate the possible impact on code reusability.

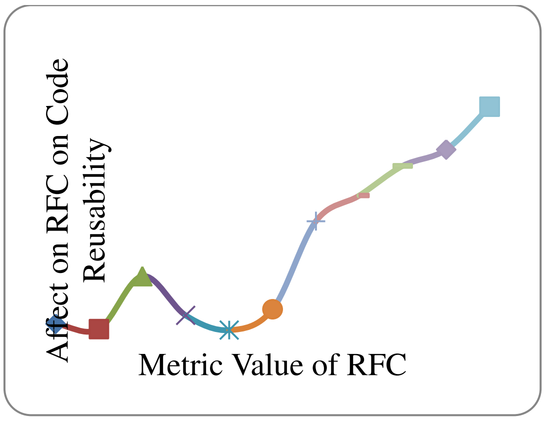

Figure 3. Affect of RFC on Code Reusability (Synthetic)

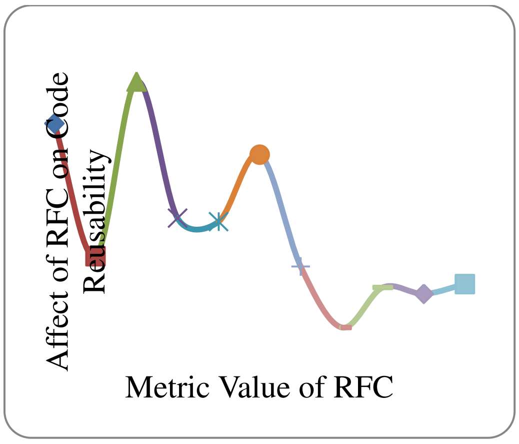

Figure 4. Affect of RFC on Code Reusability (Original)

Figure 3 shows the outcome of the synthetic data where the parameters are kept in random state, which is less likely to happen in complex software projects. Although the outcome shows increase in code reusability, but there could be possibly latent perspective of the outcome. Hence, we use original data from our ERP and SCM project where RFC parameters (α and β) were structured in increasing order. The outcome shows that there is a significant drop in code reusability. Therefore, the outcome stated in Figure 4 is within the agreement of the hypothesis that "*The complexity in code design and reusability decreases for maximized outcomes of RFC*".

8.3. Effect of NOC on Code Reusability

Figure 5 highlights the effect of NOC over code reusability. The outcome shows that code reusability have significantly improved with the numerical values mentioned in Table 2. A closer look into the numerical values will show that parameters (α and β) are in same trend of maximization order. Therefore, overall it can be said that increase in NOC has resulted in improved code reusability. The prime reason behind this is larger values of NOC will represent maximized amount of base classes that results in significant code reusability. Therefore, the null hypothesis stating "*Code reusability improves for increasing values of NOC present in class*" is found accepted.

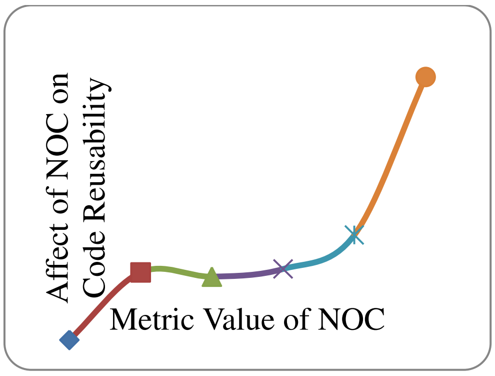

Figure 5. Affect of NOC on Code Reusability

8.4. Effect of CBO on Code Reusability

It is said that higher values of CBO is not good for software engineering as excessive coupling can turn the design evaluation quite difficult and complex. But, software projects using object oriented may more likely have CBO values than expected. Hence, it is found that proposed system can drastically enhance the code reusability even if the software projects do have higher CBO values. Figure 6 shows that with increasing CBO values for any forms of object oriented codes, it is feasible to get more values of code reusability. Hence, the hypothesis stating that "*Increased CBO values doesn't have much impact over code reusability*" is accepted.

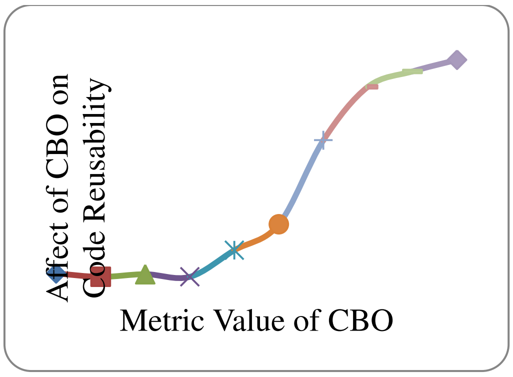

Figure 6. Affect of CBO on Code Reusability

8.5. Effect of WMC on Code Reusability

Theoretically, it is expected that increased values of WMC may result in design complexity. However, when we performed our evaluation using proposed technique, we found no much adverse effect on code reusability for certain initial rounds. A closer look into the outcomes shown in Figure 7 will highlight that although code reusability increases due to exposure to various methods used over the classes, code reusability increases, but at the same time, the performance is found to be degrading for further increment in (α and β) parameters of WMC. Hence the null hypothesis stating "Code reusability declines for increasing values of WMC in every class" is found accepted in proposed system.

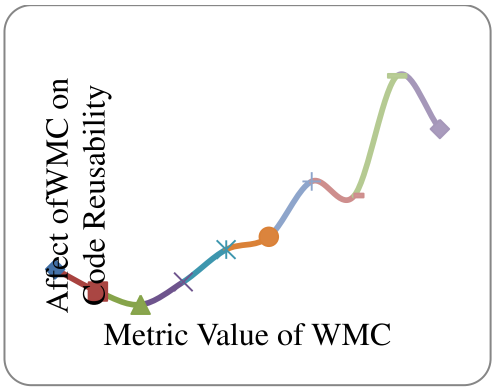

Figure 7. Affect of WMC on Code Reusability

A closer look into the cumulative outcome of the study will show that code reusability increases for some CK metric parameters and degrades for some other CK metric parameter. Our observation encounters more processing and analysis time to carry out testing. Our work was in the direction of enhanced cohesion between the significant methods as well as to ensure that there is a minimal coupling between the potential objects. This condition has ensured to retain better code reusability by ensuring higher cohesion among the classes. If the intensity of the coupling is more than we found such cases to be quite non-supportive of code reusability. Moreover, it is also explored that inclusion of more number of methods in ERP projects causes the design class to incur more computational complexity and thereby leads to inferior design patterns. Internal callbacks and communication for message more than 150 causes degradation in the code reusability.

From the result discussed in this section, it can be seen that there are various factors of CK metric that directly impacts the code reusability process. During working on new set of the code, it is essential that new components and classes designed should have to be within the anticipated outcomes to claim for reusable codes. Hence, our proposed technique can presents a framework that can be used to measure the relationship between the CK metric and code reusability. The design to be incorporated for the new set of the code must have permissible limit of code reusability, which the designer can easily set up during the formal verification process. The proposed system is highly extensible for various forms of software projects other than ERP and SCM.

Table 2. Numerical Outcomes of Study

	ERP Project (C: 220, MC: 120)				SCM (C:570, MC: 380)			
Metric	Value	α	β	C_r	α	β	C_r	Avg
DIT	1	26	3	11.54	32	3	9.38	10.46
	2	36	9	25.00	59	15	25.42	25.21
	3	63	35	55.56	84	46	54.76	55.16
	4	33	24	72.73	121	90	74.38	73.55
	5	36	31	86.11	132	113	85.61	85.86
	6	18	18	100.00	112	110	98.21	99.11
RFC	5	11	1	9.09	3	1	33.33	21.21
	10	12	2	16.67	9	2	22.22	19.44
	20	8	2	25.00	4	2	50.00	37.50
	34	27	4	14.81	6	2	33.33	24.07
	48	7	1	14.29	21	5	23.81	19.05
	65	5	1	20.00	34	11	32.35	26.18
	85	13	7	53.85	63	37	58.73	56.29
	100	41	26	63.41	81	54	66.67	65.04
	140	25	18	72.00	97	76	78.35	75.18
	165	31	24	77.42	119	100	84.03	80.73
	200	33	31	93.94	92	89	96.74	95.34
NOC	0	209	1	0.48	530	1	0.19	0.33
	1	3	1	33.33	2	2	100.00	66.67
	2	4	1	25.00	3	3	100.00	62.50
	3	5	2	40.00	4	4	100.00	70.00
	4	5	2	40.00	3	5	166.67	103.33
	6	6	1	16.67	1	5	500.00	258.33
CBO	1	3	1	33.33	9	1	11.11	22.22
	3	9	2	22.22	10	2	20.00	21.11
	4	7	2	28.57	13	2	15.38	21.98
	5	18	4	22.22	15	3	20.00	21.11
	8	42	11	26.19	21	7	33.33	29.76
	10	23	8	34.78	60	25	41.67	38.22
	14	27	18	66.67	76	49	64.47	65.57
	19	24	21	87.50	88	69	78.41	82.95
	21	34	31	91.18	131	111	84.73	87.95
	24	31	29	93.55	119	107	89.92	91.73
WMC	3	13	4	30.77	2	1	50.00	40.38
	5	20	4	20.00	5	2	40.00	30.00
	8	22	5	22.73	16	4	25.00	23.86
	10	15	5	33.33	28	10	35.71	34.52
	14	22	11	50.00	57	28	49.12	49.56
	18	23	13	56.52	88	48	54.55	55.53
	30	15	10	66.67	51	49	96.08	81.37
	50	37	28	75.68	69	51	73.91	74.79
	80	24	30	125.00	44	60	136.36	130.68
	100	31	38	122.58	184	164	89.13	105.86

8.6. Outcome Assessment

In order to validate the proposed system, we have developed a simple assessment model. The prime theme of this validation model is to understand the effect of increasing uncertainty in the 3 types of software projects towards code reusability as shown in Figure 8. The assessment model is designed over Matlab which takes the empirical values of software metric suite from Customer Relationship Management project, Supply Chain Management project, and Enterprise Relationship Management project. Using the introduced equation of code reusability, the system performs computation. The outcome of the code reusability will be subjected to various levels of uncertainties. We define uncertainties as various hidden parameters like skill gap, requirement volatility, ignorance of complying with software development life, project slippage etc, which is beyond control of any human. We like to understand that in case of hidden or unforeseen circumstances in any project management team, what is the performance of code reusability in that case? Hence, we apply mathematical approach in this regard by using Markov model [36]. Applying Markov modelling, it becomes possible to map all the real-time uncertainties into an empirical parameter and apply it over the proposed code reusability model to understand its behavior.

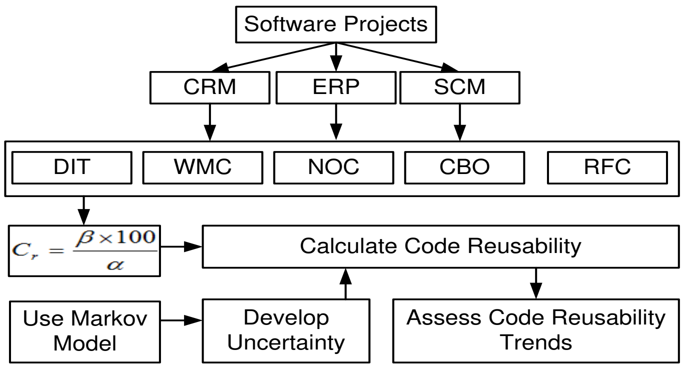
Software Projects
CRM
ERP
SCM
DIT
WMC
NOC
CBO
RFC
$C_r = \frac{\beta \times 100}{\alpha}$
Calculate Code Reusability
Use Markov Model
Develop Uncertainty
Assess Code Reusability Trends

Figure 8. Model for Assessing Proposed System

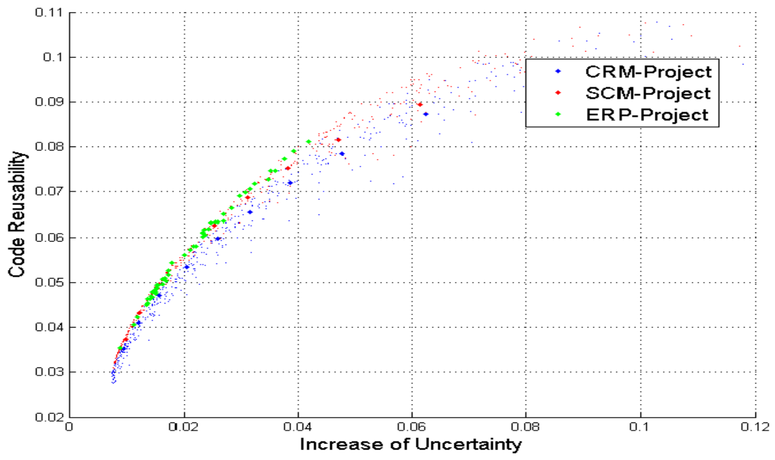

Figure 9. Comparative Analysis of Software Projects

Figure 9 shows the comparative analysis of the three different software projects based on object oriented designs using a scattered diagram. The outcome shows dominancy of ERP projects, where the code reusability is found to be quite significantly increase with increase of uncertainty. However, the adverse side of this outcome is that such dominant result is only visible till 0.01-0.04 levels of uncertainty values. Better than ERP project, SCM project was found with sparse but increasing values of code reusability. A closer look into the scattered plot will show that there is a significant increase in code reusability from 0.01-0.08 values of uncertainties. However, owing to inclusion of maximum number of classes and methods, the values of DIT and WMC increases resulting in design complexities. This is the main reason why code reusability for SCM can be evaluated in sparse manner and in slower pace, but with better accuracy compared to ERP projects. We have also testified the assessment model with a middle-sized CRM project. Normally, the amount of design complexities CRM projects are highly increased in multifold. However, using the proposed equation, we testified our hypothesis and found that code reusability to be significantly enhanced for CRM project inspite of massive design complexities involved in project architecture.

9. Conclusion

At present, we have drawn relationship between the most standard software metrics and code reusability. We have testified it on three complex software projects of object-oriented designs and found that our model can significantly calculate code reusability for any extent of complexities even it is very much uncertain. Using mathematical and stochastic approach of Markov Modelling, we proved that our model can extract more data of code reusability on increasing uncertainties. Design pattern plays an important role in software engineering. With the increasing demands of the customers, the IT industries and software project developers are increasingly seeking consultation to minimize the cost of production from more than a decade. In the form of various cost cutting procedures, code reusability is the more prominent one and requires a highly skilled technical architecture to take a decision. A code reusability deals with two challenging aspects i.e. i) deciding which part of the code to be retained same and ii) deciding which part of the code will be need to be designed from scratch. In the first challenging aspect, a developer can easily decide about what part of the code will be required to be retained based on the client's requirement. However, the difficult part is to make a decision related to the new code that is required to be built from base. Normally, depending on an experienced architecture, the new set of the code that needs to be programmed is designed in such a way that it should posses certain level of code reusability for the future client, which is unpredictable. An unpractical design in this case will go for complete loss of production and may not meet the reusability factor for new projects. Hence, our future direction of study will focus on estimating the level of code reusability for complex software projects. We anticipate that our design concept will highly encourage and motivate the stakeholder to consider it as most cost-effective tool till date.

References

[1] J. Mishra, A. Mohanty , "Software Engineering", *Pearson Education India, Electronic books,* pp. 387, 2011

[2] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, "*Experimentation in Software Engineering: An Introduction*", Springer Science & Business Media, pp. 204, 2012

[3] S. K. Dubey, A. Rana, A Comprehensive Assessment of Object-Oriented Software Systems Using Metrics Approach, *International Journal on Computer Science and Engineering*, vol. 02, no. 8, pp.2726-2730, 2010

[4] N. Mohammed, A. Govardhan, Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development, *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 6, 2011

[5] B.Jalender, A.Govardhan, P.Premchand, Designing code level reusable software components*, International Journal of Software Engineering & Applications (IJSEA)*, vol.3, no.1, 2012

[6] "Classification of Software Metrics in Software Engineering", http://ecomputernotes.com/software-engineering/classification-of-software-metrics, Retrived, 10th Dec, 2012

[7] M. Sarker, "An Overview of Object Oriented Design Metrics", *Master Thesis Department of Computer Science*, Umea University , Sweden, 2005

[8] "Metamil", http://www.metamill.com/, Retrived, 10th Dec, 2015

[9] "Sourceforge", metrics.sourceforge.net, Retrived, 10th Dec, 2015

[10] "Codeswat Custom Solutions", http://codeswat.com/, Retrived, 10th Dec, 2015

[11] J. Alghamdi, R. Rufai, and S. Khan. Oometer: *A software quality assurance tool. Software Maintenance and Reengineering,* 2005. CSMR 2005. 9th European Conference, pp. 190–191, 2005

[12] V. R. Basili, L. Briand and W.L. Melo, "A Validation Of Object-Oriented Design Metrics As Quality Indicators", *Technical Report, Univ. of Maryland, Dep. of Computer Science*, College Park, MD, 20742 USA. April 1995.

[13] C.N.S.Anna, A.F.Garcia, C.V.F.G. Chavez, C.J.P.d. Lucena, A.V. Staa, "*On the Reuse and Maintenance of Aspect-Oriented Software:An Assessment Framework*", PUC-RioInf.MCC26/03 Agosto, 2003.

[14] [14]P.S,Kaur, and A. Singh.,"Modeling of Reusability of Object Oriented Software System", *World Academy of Science, Engineering and Technology,* vol. 56, pp.162. 2009.

[15] M. Kaur, M. Mahajan, P.S. Sandhu, "*A k-NN based approach for Reusability Evaluation of Object-Oriented Based Software Components*, International Conference on Information and Communications Security, 2011

[16] U. Kumari, S. Bhasin. Application of object-oriented metrics to C++ and Java: A comparative study. *ACM SIGSOFT Software Engineering Notes*, vol. 36(2), pp.1-10, 2011

[17] P. Edith Linda, E. Chandra and J. Sharmila, "An Approach to Evaluate Object Oriented Class Structure using Score Carding Framework*", International Journal of Software Engineering and Its Applications*, vol. 9, No. 3, pp. 9-16, 2015.

[18] D. Wu, L.Chen, Y. Zhou and B. Xu, "A metrics-based comparative study on object-oriented programming languages", *State Key Laboratory for Novel Software Technology at Nanjing University,* Nanjing, China, DOI reference number: 10.18293/SEKE2015-064, 2015.

[19] K.P. Srinivasan And T. Devi, "A Comprehensive Review And Analysis On Object-Oriented Software Metrics In Software Measurement", *International Journal on Computer Science and Engineering (IJCSE),* vol. 6, no.07, 2014.

[20] M. Scotto, A. Sillitti, G. Succi, T. Vernazza, "A relational approach to software metrics", *ACM Symposium on Applied Computing*, pp.1536-1540, 2004.

[21] S. Singh, P. Singh, N. Mohan, P.S. Sandhu, "Logistic Model Trees based Approach for Prediction of Reusability of Object Oriented Software Components", *International Journal of Research in Engineering and Technology*, vol. 1, No. 3, 2012.

[22] R. Subramanyam, M.S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects*", IEEE Transactions on Software Engineering*, vol. 29, no. 4. 2003.

[23] A. Shaik, C.R.K. Reddy, B. Manda, C. Prakashini and K. Deepthi, "Metrics for Object Oriented Design Software Systems: A Survey",*Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS),* vol. 1(2), pp.190-198, 2010.

[24] S. I. Zahara, M. Ilyas and T. Zia, "*A Study of Comparative Analysis of Regression Algorithms for Reusability Evaluation of Object Oriented Based Software Components*", International Conference on Open Source Systems and Technologies (ICOSST), 2013.

[25] H.M. Manoj and A.N. Nandakumar, "A Survey on Modelling of Software Metrics for Ranking Code Reusability in Object Oriented Design Stage", *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, Issue. 12, 2014.

[26] A. Oberoi and D. Arora,"Quality Model For Analysis And Implentation Of CK Metrics Through Neural Networks: *International Journal of Engineering Research and Applications (IJERA)* ISSN: 2248-9622.National Conference on Advances in Engineering and Technology, AET, 2014.

[27] N.Goyal and D. Gupta, "Reusability Calculation of Object Oriented Software Model by Analyzing CK Metric",*International Journal of Advanced Research in Computer Engineering & Technology (IJARCET),* vol. 3 Issue. 7, 2014.

[28] N.Jayalakshmi and Nimmati Satheesh," Software Quality Assessment in Object Based Architecture*",International Journal of Computer Science and Mobile Computing*, vol.3, issue.3, pp. 941-946, 2014.

[29] A.V. Hudli and R.V. Hoskins: "*Software metrics for OOD",* IEEE International conference, 2002.

[30] H.Lilu, K.Zhou and S.Yang: "*Quality metrics of OOD for Software development and Re-development",* First Asia-Pacific Conference on Quality Software, 2002.

[31] N. Paliwal, V.Shrivastava and K. Tiwari, "An Approach to Find Reusability of Software Using Objet Oriented Metrics*", International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, issue 3, 2014.

[32] N. Chauhan and M. V.Gupta, "Evaluation Of Metrics And Assessment Of Quality Of Object Oriented Software", *IJRET: International Journal of Research in Engineering and Technology,* vol. 03, special issue: 14, 2014.

[33] D. Gupta, V. K. Goyal and H. Mittal, Comparative Study of Soft Computing Techniques for Software Quality Model*, International Journal of Software Engineering Research & Practices*, vol.1, issue: 1, 2011.

[34] R. Alcalá, J. Casillas, O.Cordón, and F. Herrera, "Linguistic modeling with weighted double-consequent fuzzy rules based on cooperative co-evolutionary learning*", Integrated Computer-Aided Engineering*, vol. 10, no. 4, pp. 343-355, 2003

[35] "SPSS software", http://www-01.ibm.com/software/analytics/spss/, Retrieved 10th Dec, 2015

[36] M. Stamp "A revealing introduction to hidden Markov models", *Department of Computer Science San Jose State University,* 2004.