RUCM: A Measurement Model for detecting the Most Suitable Code Component from Object Oriented Repository

Sumit Jain
Acropolis Technical Campus
Indore Bypass Road, Near Tillore
Rala Mandal, Indore (M.P.) - 452020

Mohsin Sheikh
Medi-caps Group of Institutions
A.B. Road, Pigdamber Rau
Indore (M.P), India – 453331

ABSTRACT

Software quality based applications development is the main concern is user satisfaction. It increases the reliability and efficiency of information retrieval and management. As the bundle of code created day by day the repository storing such code is regularly migrates the older code in to legacy systems. To develop and facilitate new object oriented model based application with improved problem solving capabilities such code has to be re-factored and reused effectively. The legacy systems have the collection of both the types of the code: procedural and object oriented. The procedural code is converted into object oriented code by using the phenomenon of re-engineering and the object oriented code database is searched for reusable code components. Thus to make the effective and timely detection of such reusable components tools is required. All the existing tools for such detection use various metrics for measuring and analysis of compatibility, price and development effort required to re-engineer those components. Also the current system will only focuses on using cohesion and coupling based metrics. But accuracy is the problematic issues in all of them because of their few metrics usage conditions.

This work proposes a novel RUCM (Reusability Utility Count Model) for analyzing the reusability value. It takes various key features of code for calculating the above. The work focuses on satisfying the quality attributes by applying all the modularity principles in metrics design and measurement. To do that effectively this work had developed six composite metrics: LOC, LMD, MD, DOC UOS, and IC. In its primary work level the proposed approach seems to provide effective results in near future.

Keywords

RUPM (Reusability Utility Count Model), Object Oriented, UOS (Understandability of Software), IC (Interface Complexity), DOC (Degree of Cardinality), LMD (Low Modification Degree), ALOC, MD (Modularity Degree), Cohesion, Coupling;

1. INTRODUCTION

We Object oriented programming is accepted widespread due to its powerful mechanism which provides strong, robust, flexible and easy design which helps in the maintaining program. It uses the modular concept of creation of class instead of taking the whole program as complete unit. It comprises of encapsulation, polymorphism, inheritance, exception handling, and modular approach and must follow the component based software engineering phenomenons. Some useful entities like cohesion and coupling had also taken into account. The ability to achieve high reuse levels has been especially described to object oriented software development [1]. Reusability is always working as a component of refactoring and has various approaches to solve the decidability of such components utilization in better manner. Few of them are mining based like association rule, classification and clustering; some of them support the logic driven trained identification come under neural network like perceptions, MLP etc while others are generalized. All of the above solution domains will provide the effective identification of reusable codes but always required different metrics for analyzing such behaviors. These software metrics are used for the structural analysis of the different procedures and user interfaces (UI) [2].

Code reuse is the most active and creative concept to improve in the implementation of the newer software. It can measure the degree of features that are reused in guiding application as well as the increases productivity and quality with a very less cost [3]. Code reuse is the use of existing software component to build new software system. This factor in new development process is highly dependent on some reusability criterion. The notion of reusability is find out the most reusable software component from the pool of existing software component. A huge amount of research work has been done to identifying the quality of reusable components based on metrics on the design level but there is very less work on the framework that makes use of these metrics to find reusability of software components at implementation level. So a few amount of research are remains to find out the reusability of software component (programming code) at implementation level. We determined from requirement phase to the maintenance phase of the software development. In Software Development Life Cycle (SDLC) reuse concept is not determined to only coding stage. These are several phases where software development is reused: such as Code, Requirement, Architecture / Design Documentation, Test Plans, Specifications, Design, Manuals, Templates and Design Decisions. So many other application areas such as Neural Network, Soft Computing and AI are continuously accepting its utility in their domains [4].

Out of the above mentioned domain this work focuses its direction towards the code reusable components to reduce the development efforts and cost. It is a metrics based approach which gives us accurate reuse measurement of code. It helps to identify the reusability of any object oriented code, which helps in various organizations and industries that they can choose the most reusable module from the pool of existing software modules. This tool accepts the object oriented code as an input, applies reusability matrices on the code, measures

the reusability and returns the reusability factor as an output. Now the developers compare these reusability factors of the existing software component to find out the most reusable code to reuse in the further software development process. This is an efficient and most effective approach to identify the reusability of object oriented code.

This paper gives a brief study on reusability detection of existing system and after analyzing various attribute it proposes a novel RUCM method to overcome the issues. It gives an accurate object oriented based detection of components satisfying the suggested metrics for more accurate and effective results. This paper is divided into three subsections.

2. BACKGROUND

Now a day in software industry increasing the productivity is taken as a major task which reduces the effort, cost and time while parallel raises the quality levels. Maintainability, accountability, reliability robustness availability and reusability are the concerning quality parametric attributes. For the large software's problem development and solution identifications is taken in advancements of various novel technological variables. Thus it has to be modularized for better understanding and hence uses component based software development model (CBD) [5]. To achieve the bigger goal in smaller time and effort reusability is used. It increases the productivity, which is to reuse the existing component, because a lot of time and effort already have expended for the developed product. Old software component that is well tested and designed already, that one should reuse the being software [6]. Many times modules are not developed for reusable extends to highly product development time and cost. Thus, one should explore that which existing component or module is more suitable for reuse, and try to reuse. This work proposes a novel Reusability Utility Count Model (RUCM) for calculating reusability for the object oriented programs. By using this model one can be resolve that whether the existing component is suitable for reuse or not. It uses various metrics to identify its suitability for new development.

Specifically the work uses only object oriented phenomenon's thus the metrics will also reflect the same for complexity calculations for both internal and external value measurement. There are two methods for reuse of code: first is to develop the reusable code from scratch or identify and extract the reusable code from already developed code [7]. The cost of making the software from older code repository components can be saved by identifying and extracting the reusable methods from developed software systems. Development of large scale development requires fewer changes in legacy systems to improve the applicability of any existing module into new development. It will be given as:

- Apply the re-factoring and re-engineering to legacy modules of existing systems.
- Identify the most suitable component based on internal and external quality metrics.
- Merge the newly identified module with current methods.
- Check the module behavior after integration before releasing it.
- Make the simultaneous changes in existing implementation and let the product open for future extension and up gradation.

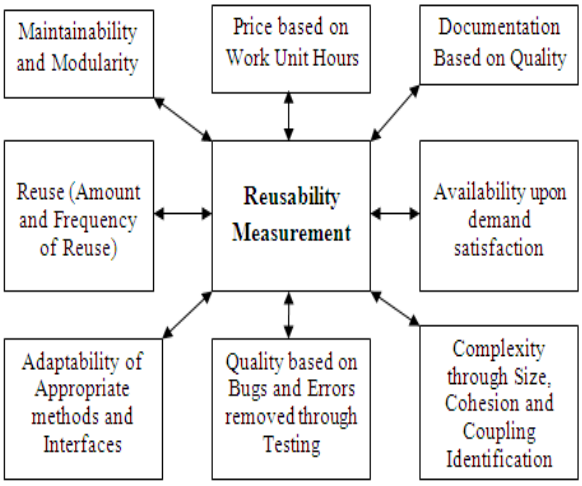

Figure 1: Reusability Measurement Seven Step Model

Reusability Measurement: is a multi step models which includes identification of modularity and maintainability in existing modules. It gives the details about the number of modules for a specific process. Modularization is used to analyze the current state of reusable components. This component is analyzed to detect the changes required for suitability with developing project by which price and work unit hours can be calculated to identify the cost elements. Most reused items give the frequency of reusable modules. By taking this frequency frequent methods and components can be sorted out from repository. Now after taking the frequent code the internal structure of code is analyzed using various comparative metrics like adaptive LOC, NOC etc [8]. Such adaptive changes will measures the bugs and errors which will later on removed for using it as it was. These internal structures calculate the complexity, size by using the reusability parameters like cohesion and coupling. After considering all the factors successfully the demand of developer is satisfied completely and will raise the quality levels.

3. RELATED STUDY

In the paper [9], the author proposes a new mathematical model based on vector analysis model, where the requirements are posed as Vectors in an N-Dimensional Space to measure reusability of software design in terms of reusability of the use cases. It also uses the vector model, measure Complexity Reduction Factor (CRF) which gives an effort and hence in cost with increasing reusability. It is the important phase of reengineering. The work had also used them as orthogonal vectors, each aligned along a particular dimension and having a magnitude equal to the development complexity i.e. the Unadjusted Use Case Weight (UUCW).This factors is analyzed as an quantitative measures and gives an considerable reduction in efforts.

A novel multi-objective optimization approach for improving existing packages structure is given in [10]. It gives an optimization approach which aims at increasing the cohesion and reducing the coupling and cyclic connectivity of packages, by modifying as less as possible the existing packages organization. It also mentions several constraints to guide the optimization process with regard to extra design factors. For this the authors uses the Non-Dominated Sorting Genetic Algorithm (NSGA-II). The paper will evaluate the optimization mechanism through an experiment layer.

Various surveys had also taken during the last few decades to resolve the reusable components issues. Among them the paper [11] proves as a pivot stone. This article gives a survey metrics and models of software reuse and reusability and provides a classification structure. The metrics studied in this paper is: cost-benefit models, maturity assessment models, amount of reuse metrics, failure modes models, reusability assessment models, and reuse library metrics. They can be used to demonstrate the value of the library to management as well as to provide information for continuous quality improvement. Later on the survey is extended by [12] using the object oriented metrics. In this the author also suggested a mathematical model of using those in exact representation. The metrics studied includes aggregation across metrics, usage across metrics, equivalent formulation of metrics by multiple researchers, and exploitation of traditional metrics for object-oriented metrics. The paper traces the chronological development of research in this area, and uncovers gaps that suggest opportunities for future research.

In the paper [13], the author proposes a novel classification framework (TAPROOT). This was defined across two independent vectors: category (design, size, complexity, reuse, productivity, quality) and granularity (method, class, system). The proposed metrics refer to abstractions of the object-oriented paradigm. At the initial level of work the suggested metrics seems to cover each aspects of reusable components detection. But the requirement specific reusability is not mentioned in this. To solve the above issue the paper [14] suggest a structured reusability requirements model. It uses the well-accepted Goal-Question-Metric (GQM) paradigm along with ad-hoc phenomenon, for deriving appropriate measurements and metrics for software reusability. The purpose here is to improve, the issue is reusability assessment, and the objects are software components. The outcomes of the paper taken as: the reusability of the software component depends on various non-functional characteristics while fulfilling functional requirements. Some of the language based analysis and transformation form procedural oriented to object oriented metrics based on class and methods is studied in [15]. It uses various metrics to assess the space and time performance on different legacy modules.

The above addressed research of discovering the new object oriented metrics had further extended by various other authors. In the paper [16], two metrics are proposed for measuring amount of generality included in the code and then analytically evaluated against Weyuker's had suggested the set of nine axioms for this problem. This set of metrics is then applied to standard projects and accordingly ways in which project managers can use these metrics are suggested. A primary stage gives the great results. But quality parameters remain unaddressed which is later on solved by using HALSTEAD metrics and SQA analysis [17]. The paper also gives a comparison on various other metrics like CK metrics, Moose Metrics, QMOOD Metrics, GQM, MOOSE, and EMOOSE.

So for measuring the reusability of code component metrics based approach is most suitable. So many existing metrics is used to identify the quality of those components. Some of them had extended like Chidamber and Kemerer in [18], and neural network based approach in [19]. Different algorithms have been experimented and results are recorded in terms of Accuracy, Mean Absolute Error (MEA) and Root Mean Square Error (RMSE). Some of the authors also worked on independent metrics such as Number of Template Children (NTC), Depth of Template Tree (DTT) Method Template Inheritance Factor (MTIF) and Attribute Template Inheritance Factor (ATIF), to measure the reusability for object-oriented systems [20]. The above mentioned metrics is extended by addition of few more accuracy measurable factors like WMC, CBO and LCOM using CK metrics based approach [21].

In the paper [22], the author describes an efficient algorithm that performs a fine-grained analysis of cycles among application packages. The paper proposes multiple metrics to rank cycles by their level of undesirability, prioritizing cycles that are the more undesired by developers. It retrieves a set of short cycles that covers all dependencies of the SCC. It has a polynomial time and space complexity. One metric, called diameter is based upon the distance between packages involved in the cycle. For each program, it computes and rank the cycles. To that extent, it counts how many cycles in the k first ranked by the algorithms are undesired, and how many of the k last cycles are desired. It also gives a comparison of these multiple ranking metrics on four large and mature software systems in Java and Smalltalk.

4. PROBLEM IDENTIFICATION

To assess the reusability using existing measures can be achieved by analysis the principles of modularity in legacy systems. Taking object oriented approach as a work orientation changeability, information hiding and reusability needs to be calculated. It is used to calculate the dependencies between the inter and intra packages of the class like inheritance. Hence the work is selecting the most suitable reusable components from the huge database of repository after applying some principles as given in [23]. Thus the work should follow following principle:

- Modularity Principle: Only the module interfaces are accessible by other modules
- Hiding Information and Encapsulation Principle: The communication between the packages should be as little as possible. Thus, the number of methods/implementations/classes that a package exposes to other packages should be relatively small.
- Changeability, Maintainability and Reusability: While making changes in a given module, the propagation of changing impact on other modules should be minimal. In the same context, modules should be reusable pieces of software. A module should interact with other modules via well identified interfaces and requires identifiable services from other modules
- Commonality-of-Goal vs. Similarity-of-Purpose: A module should provide particular services to other modules. Therefore, the programs inside a module should have a common goal, which should be: capturing the module design decisions and implementing the module services.

Thus the work must require the proper identification of reusable components and must follows the above guiding rules for improved detection. The satisfiability of these principles is calculated by detailed analysis of various existing metrics using mathematical properties. The core objective of this work is to identify the relative modularization using existing metrics. Detecting the change in configuration of code components will leads us to effective maintenance of legacy modules. It needs to calculate the extent of package dependencies

If there are large number of function and class templates in a system, the testing and debugging of the function and class becomes more complicated since it requires greater level of understanding at the part of developer.

5. PROPOSED RUC MODEL

Many different metrics have been proposed for object-oriented architecture. The object-oriented metrics that were chosen measure principle structures that, if they are improperly designed, nasty effect on the design and code quality attributes. The selected object-oriented metrics are primarily applied to the concepts of classes, coupling, and inheritance. For some of the object-oriented metrics discussed here, multiple definitions are given. As with traditional metrics, researchers and practitioners have not reached a common definition or counting methodology. In some cases, the counting method for a metric is determined by the software analysis package being used to collect the metrics. Reusability evaluation System for function Based software Components can be framed using following steps:

Step 1: Selection of Metric Suit for Procedure Oriented Paradigm: A framework of metrics is proposed for structural analysis of procedure or function oriented software. The code of software is parsed to calculate the metric values.

The following suits of metrics are able to explore different structural dimensions of procedure oriented components.

Step 2: Calculate the metric values of the sampled software components.

Step 3: Reconfiguring the solution structure to improve the possibility of using predefined components available at the next phase.

Step 4: Acquiring, instantiating, and modifying predefined components.

Composite Metrics Used

1) Understandability of Software (UOS):

a. It gives the clearness of the existing code for its reuses. This metrics will identify the commented percentage of code which increases understandability. It is used to evaluate the attributes of Reusability, Understandability and Maintainability. CP is computed by number of comment lines separated along Line of Code.

b. Comment Percentage COM (%) = Total Number of Comment Lines (CLOC) / Total Lines of Code

2) Interface Complexity (IC):

It is used to identify the number of factored modules satisfying the quality attributes using McCabe cyclomatic complexity. The Cyclomatic complexity is not used for class because of inheritance and interface. It can be used for individual methods but collectively with other measures can be used for class. It uses three basic metrics:

1. WMC: Weighted Method per Class is used measures the complexity of an individual class. Classes with large numbers of methods are likely to be more application specific, limiting the possibility of reuse.

Method 1=∑Individual Methods Complexity (M1, M2….Mn)

Method 2=Assigns Fixed Complexity to each (1) then ∑Individual Methods Complexity (M1, M2….Mn)

2. DIT: Depth of Inheritance is used to define the length of the longest path of inheritance ending at the current module. The deeper a class is within the hierarchy, the greater the number of methods it is likely to inherit making it more complex to predict its behavior due to the interaction between the inherited features and new features, but the greater the potential for reuse of inherited methods.

∑Total Inheritance Level (Inherited Class Path1 +Inherited Class Path2…..Inherited Class Pathn)

3. NOC: Number of Children is the number of immediate subclasses subordinate of a class in the hierarchy. The greater number of children, the greater the likelihood of improper abstraction of the parent and may be a case of misuse of sub-classing. But the greater the number of children, the greater is the reuse capability. NOC, therefore, primarily valuates testability and design.

3) Degree of Cardinality (DOC):

It is also called as response for a class (RFC). The RFC is the cardinality of the set of all methods that can be invoked in response to a message to an object of the class or by some method in the class.

DOC= ∑ (Class Method 1 to n+ Distinct Method Invoked Directly)

The larger the number of methods that can be invoked from a class through messages, the greater the complexity of the class. Its larger value requires a greater level of understanding and hence lower will be the reusability.

4) Modularity Degree (MD):

This MD is used to calculate the degree of modularity of the program and gives a percentage value which defines the total number of expected modules. It uses its two supportive metrics LCOM (Lack of Cohesion Method) and CBO (Coupling between the Object). LCOM must be high and CBO must be less for better reusable component.

a. LCOM measures the degree of similarity of methods by an instance variable or attributes. It calculates each data field in a class what percentage of the methods use that data field. Average the percentages then subtract from 100%. Lower percentages mean greater cohesion of data and methods in the class. Secondly, count the number of disjoint sets produced from the intersection of the sets of attributes used by the methods.

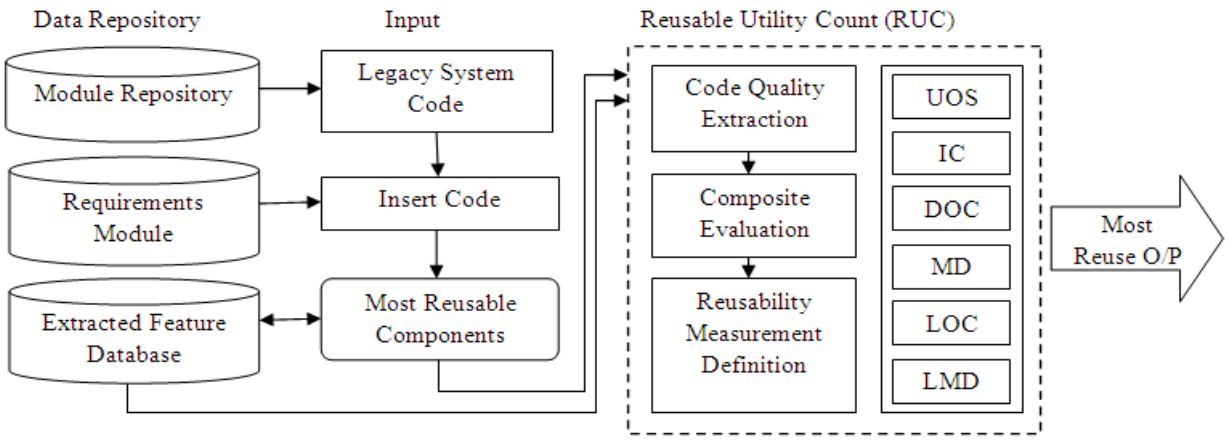

Fig 2: Proposed Design Architecture of RUP Model

b. CBO is a count of the number of other classes to which a class is coupled. Coupling is a measure of the strength of association established by a connection from one entity to another. It is measured by counting the number of distinct non-inheritance related class hierarchies on which a class depends. Excessive coupling is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is reused in another application. The larger number of couples, the higher the sensitivity to changes in other parts of the design and therefore maintenance is more difficult. Strong coupling complicates a system since a module is harder to understand change or correct by itself if it is interrelated with other modules. Complexity can be reduced by designing systems with the weakest possible coupling between modules

5) Size (LOC):

All physical lines of code, the number of statements and the number comment lines. However, since size limitations are based on ease of understanding by the developers, routines of large size will always pose a higher risk in attributes such as Understandability, Reusability, and Maintainability. This metric can be used to evaluate all the attributes, but most often is a measure of Understandability, Reusability, and Maintainability.

6) Low Modification Degree (LMD):

It can be measured by analysis the code which is change a countable number of times. More the code lines is modified assures that the sample template is reusable or not. It measures the total function template used and code modified adaptively. It measure in metrics form is defined as the PCM (Percentage Code Modified) value. It increases the customizability of the code and gains user attraction.

Low Modification Degree, LMD, as follows:

LMD = COM + CBO - (Delta*MD)

Proposed Formula (Apply on each class of the code)

Reusability of a class = a*(COM)-b*(WMC) +c*(DIT) +e*(NOC) +f*(DOC) +g*(LCOM) –h*(CBO) -i*(LOC) +j*(PCM)

Where a, b, c, d, e, f, i, j are empirical constants.

6. EVALUATION PARAMETER

Calculation of reusability from an existing code component is a difficult task because it has various quality attributes. To measure its compatibility with current developed module the identified code component must be interoperable and easy to understand. Along with above characteristics the proposed work has to satisfy following object oriented consideration principles:

Rule 1: Deeper a particular class is in the hierarchy, the greater the potential for reuse of inherited methods. It states that reusability of a class increases with increase in DIT of a class. So DIT has positive impact on reusability of a class.

Rule 2: A moderate value for NOC indicates scope for reuse. Up to particular threshold value NOC has positive impact on reusability of a class.

Rule 3: Excessive coupling indicates weakness of class encapsulation and may inhibit reuse. It indicates that coupling has negative impact on reusability of a class.

Comparison Criteria: It is made on the basis of value of MAE, RMSE and Accuracy values of the proposed model. The details of the MAE and RMSE are given below:

- Mean absolute error (MAE):

Mean absolute error, MAE is the average of the difference between predicted and actual value in all test cases; it is the average prediction error.

- Root mean-squared error (RMSE):

RMSE is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. The conclusions are made on the basis of the results calculated in the previous section.

7. CONCLUSION

Calculation of reusability in software module is taken as feature extraction and quality satisfaction problem. Code repository holds all the legacy components and software analyst needs to ally its reusability identification mechanism to make a clear selection. Weaker identification leads to poor reusable module selection and hence newly developed tools is required to make to above requirement feasible. The main objective of this work is to make the effective and easy identification of reusable component from the code repository. The work is also measuring reusability of procedural components to make them used in object oriented paradigm rule. Thus it needs to satisfaction and analyze the internal and external code structure completely.

The reusability leads to raise the developer's productivity which directly reduces cost and time. The work offered in this RUPM approach can calculate the reusability of any object oriented software module by using six different composite metrics. A new mathematical evaluation formula had also developed to prove the effectiveness of the approach. At the initial level of this work the proposed mechanism is providing good and satisfactory results which can be measured for any Object Oriented Code. An interface can be developed to demonstrate the working of proposed approach through which evaluation of most and least reusable module is detected. The applicability of this work can be used in any organization or firm to improve their productivity and quality.

Future Work

Some problems and concepts that remain unaddressed can be performed in future. As the reusability calculation is an accuracy based findings thus in future some more new metrics can be discover for better results. Also the concept above in this paper is for all types of Object Oriented Languages like (.NET, JAVA, CPP etc.) and can be extended for using it with procedural oriented languages also. Currently the work had applied this for all languages that support Object Oriented Code.

- Dependability and understandability of the code can be calculated in real time to decrease the delays.
- Different cohesion or coupling metrics can be extended to calculate more parameters like flexibility, maintainability, understandability and adaptive size.

8. REFERENCES

[1] Andreas S, Evigoni D Reiner D, Erik F and Micheal W, "Conception and Experience of metrics based software reuse in practice", Published in International Workshop of Software and Maintenance (WSM99), Sep 1999. pp 178-189

[2] Benjamin Van Ryseghem, Stephane Ducasse and Johan Fabry, "A Framework for the Specification and Reuse of UIs and their Models", Published in International Workshop on Smalltalk Technologies (IWST 12), ACM 2012.

[3] Yogesh Singh, Pradeep Kumar Bhatia and Omprakash Sangwan, "Software Reusability Assessment Using Soft Computing Techniques", Published in ACM SIGSOFT Software Engineering, doi: 10.1145/1921532.1921548, Vol. 36 No. 1, Jan 2011.

[4] Sonia Manhas, Rajeev Vashisht and Reeta Bhardwaj, "Framework for Evaluating Reusability of Procedure Oriented System using Metrics based Approach", in International Journal of Computer Applications, ISSN: 0975 – 8887, Vol. 9, No.10, Nov 2010.

[5] Santonu Sarkar, Avinash C. Kak and Girish Maskeri Rama, "Metrics for Measuring the Quality of Modularization of Large-Scale Object-Oriented Software", in IEEE Transaction on Software Engineering, ISSN: 0098-5589,doi:. 10.1109/TSE.2008.43, Vol. 34, No. 5, Sep 2008.

[6] Shamsher Singh, Pushpinder Singh and Neeraj Mohan, "Identification of Object Oriented Reusable Components Using Multilayer Perceptron Based Approach", in International Conference on Computer Engineering and Multimedia Technologies (ICCEMT'2012), Sep 2012.

[7] Nasib S. Gill, "Reusability Issues in Component-Based Development", in Department of Computer Science & Applications, M.D. University, Rohtak, Haryana (India).

[8] Karine Mordal, Nicolas Anquetil and, Jannik Laval, "Software Quality Metrics Aggregation in Industry", in Journal of Software Maintenance and Evaluation: Research Practice, doi: 10.1002/smr, 2010.

[9] Dipankar Majumdar, Sabnam Sengupta, Ananya Kanjilal and Swagata Kundu, "A Mathematical Reusability Model for Quantifying the Reduction in Development Effort", in ACM SIGSOFT Software Engineering Notes, doi: 10.1145/1811226.1811234 Vol 35 No 4, July 2010.

[10] Hani Abdeen, Houari Sahraouiy, Osama Shata, Nicolas Anquetilz and Stephane Ducasse, "Towards Automatically Improving Package Structure While Respecting Original Design Decisions", in Research Grant NPRP grant #09-1205-2-470 from the Qatar National Research Fund, Qatar University, Qatar.

[11] William Frakes and Carol Terry, "Software Reuse: Metrics and Models", in ACM Computing Surveys, ISSN: 0360-0300/96/0600–0415, Vol. 28, No. 2, June 1996.

[12] Sandeep Purao and Vijay Vaishnavi, "Product Metrics for Object-Oriented Systems", in ACM Computing Surveys, ISSN: 0360-0300/03/0600-0191, Vol. 35, No. 2, June 2003, pp. 191–221.

[13] Fernando Britoe Abreu and Rogério Carapuça, "Candidate Metrics for Object-Oriented Software within a Taxonomy Framework", in Journal of Systems and Software, Vol. 26, No. 1, North-Holland, Elsevier Science, July 1994.

[14] Danail Hristov, Oliver Hummel, Mahmudul Huq and Werner Janjic, "Structuring Software Reusability Metrics for Component-Based Software Development", in International Conference on Software Engineering Advances, IARIA, ISBN: 978-1-61208-230-1, 2012.

[15] Philip Newcomb, "Reengineering Procedural Into Object-Oriented Systems", in IEEE, ISSN: 0-8186-7111-4, 1995.pp 237-249

[16] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur and Ruchika Malhotra, "Software Reuse Metrics for Object-Oriented Systems", in Conference on Software Engineering Research, Management and Applications (SERA'05) By IEEE, ISSN: 0-7695-2297-1/05, 2005.

[17] Amit Sharma, Sanjay Kumar Dubey, "Comparison of Software Quality Metrics for Object-Oriented System", in International Journal of Computer Science & Management Studies (IJCSMS), ISSN (Online): 2231 – 5268, Special Issue of Vol. 12, June 2012.

[18] Johny Antony P, "Predicting Reliability of Software Using Thresholds of CK Metrics", in International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 6, June 201.

[19] Anupama Kaur, Himanshu Monga, Manupreet Kaur ,"Performance Evaluation of Reusable Software Components", in International Journal of Emerging Technology and Advanced Engineering (IJETAE), ISSN 2250-2459, Vol. 2, Issue 4, April 2012.

[20] Parul Gandhi and Pradeep Kumar Bhatia, "Reusability Metrics for Object-Oriented System: An Alternative Approach", in International Journal of Software Engineering (IJSE), Vol. 1, Issue 4, 2010.

[21] Anju Shri, Parvinder S. Sandhu, Vikas Gupta, Sanyam Anand, "Prediction of Reusability of Object Oriented Software Systems using Clustering Approach", in World Academy of Science, Engineering and Technology, 2010.

[22] Jannik Laval, Jean-Rémy Falleri, Philippe Vismara, Stephane Ducasse, "Efficient Retrieval and Ranking of Undesired Package Cycles in Large Software Systems". in Journal of Object Technology, doi:10.5381/jot.2012.11.1.a4 , Vol. 11, No. 1, 2012, pages 4:1–24.