Optimization of Component Based Software Engineering Model Using Neural Network

Gaurav Kumar[1] and **Pradeep Kumar Bhatia**[2]

Submitted in November, 2013; Accepted in September, 2014

Abstract - The goal of Component Based Software Engineering (CBSE) is to deliver high quality, more reliable and more maintainable software systems in a shorter time and within limited budget by reusing and combining existing quality components. A high quality system can be achieved by using quality components, framework and integration process that plays a significant role. So, techniques and methods used for quality assurance and assessment of a component based system is different from those of the traditional software engineering methodology. In this paper, we are presenting a model for optimizing Chidamber and Kemerer (CK) metric values of component-based software. A deep analysis of a series of CK metrics of the software components design patterns is done and metric values are drawn from them. By using unsupervised neural network- Self Organizing Map, we have proposed a model that provides an optimized model for Software Component engineering model based on reusability that depends on CK metric values. Average, standard deviated and optimized values for the CK metric are compared and evaluated to show the optimized reusability of component based model.

Index Terms – Chidamber and Kemerer (CK) metric; Component Based Software Engineering (CBSE); Neural Network (NN); Self Organizing Map (SOM).

1.0 INTRODUCTION

Reusability of software can be enhanced by using the structured approaches of Component-Based Software Engineering (CBSE). CBSE includes various object-oriented concepts such as Reusability through Inheritance, encapsulation, abstraction and polymorphism. Features of CBSE includes increase in productivity, improvement in quality, reduced time to market, broad range of reusability and effective management of complexity. The main characteristics of CBSE are:

- CBSE considers a component as a reusable entity.
- CBSE supports the development of system as the integration of components.
- CBSE provides facilities for upgrading and maintaining a system by simply changing the components that needs to be upgraded or replaced.

Software component with specified interfaces can be deployed independently and each component communicates with other component(s) by using its public interfaces. A software component is a self contained software element that can be specified formally, composed without modification according to deployment and composition standard, deployed independently from its environment without needs of other specific components.

A component interfaces provides functional properties and can be divided into 2 parts: *Behavior part* specifies behavior of a component; *Signature part* specifies operations provided by a component that are understandable by both interface provider (component) and user (other components or other software that interact with provider). A component has two kinds of interfaces that can be distinguish as *imported interface* which describes those services that a component requires from its environments, e*xported interface* which describes those services that a component provides to its environment.

Chidamber and Kemerer (CK) metric used in the proposed model includes weighted sum of all the methods in a class called *Weighted Methods per Class* (WMC); count of the number of other classes to which a given class is coupled called *Coupling Between Object classes* (CBO); in the inheritance hierarchy, length of the longest path from a given class to the root class called *Depth of the Inheritance Tree* (DIT); a count of the number of immediate child classes called *Number of Children* (NOC); count of the methods that can be invoked in response to a message received by an object of a particular class called *Response for a Class* (RFC);count of the number of classes used in the component (NC).

2.0 LITRATURE SURVEY

A brief literature review of work done in the field of Component Based Software Engineering is discussed here:

Kilsup Lee and Sung Jong Lee (2005) proposed a quantitative software quality evaluation model with respect to the Component Based Development (CBD) methodology. The evaluation is done using checklists that helps to acquire high quality software. The model proposed includes international standards (quality characteristic, sub-characteristics, quality metrics, and quality evaluation process) for software quality model and evaluation process.

Alexandre Alvaro et. al. (2006) analyzed and evaluated components using consistent and well-defined characteristics, sub-characteristics and quality attributes related metrics.

Kung-Kiu Lau and Zheng Wang (2007) classified and evaluated taxonomy w.r.t. parameters, its key characteristics based on commonly accepted parameters for Component Based Development.

Anita Gupta et. al. (2008) discussed an industrial case study involving a reusable Java-class framework and an application to use that framework. The impact of software changes on

[1,2] *Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India*
E-mail: [1]er.gkgupta@gmail.com,and [2]pkbhatia.gju@gmail.com

different development characteristics (e.g. impact of reuse and impact of refactoring) are analyzed. Perfective and corrective changes in both reusable and non-reusable software are analyzed.

Mubarak Mohammad and Vasu Alagar (2008) proposed a CBSE approach that defines the trust worthy quality attributes as first class structural elements where formalism is integrated into various stages of the development process. A development framework of comprehensive tool support and justification of their role in assuring trustworthiness during the different stages of software development is discussed.

R. Senthil et. al. (2008) described and evaluated N-tier architecture for a component based application against the external and internal quality factors to establish an enhanced component model (ECM). A case study has been carried out to establish the result that the application so developed is scalable and robust as one can migrate from one data source to another using this quality model.

Yoonjung Choi et. al. (2008) presented a Component Quality Model which includes metrics for component quality evaluation, basic guidelines for evaluations, and reporting formats of evaluations. An improvement is made in the Component Quality Model for the proper tailoring when applied in embedded system domain. Two different projects are evaluated to use as guidelines and goals for component quality improvements.

V. Lakshmi Narasimhan et. al. (2009) carried out a systematic analysis and comparison of three suites of metrics and several key inferences have been drawn from them. The metric values provided are helpful to study the behavior of metrics under various quality factors. Inferences on complexity, reusability, testability, and stability of the underlying components have been drawn from various metrics evaluations.

María A. Reyes et. al. (2009) analyzes and proposed a systematic approach that allows assessing and improving products and processes of the conceptual elements behind Component-Based Software Engineering (CBSE) for its quality evaluation and integrating the product perspective as well as process perspective.

Anju Shri et. al. (2010) used Tuned CK metric suit as input to obtain the structural analysis of Object oriented based software components. Reusability of object oriented software components is evaluated using a hybrid K-Means and Decision tree approach. The proposed reusability model produces high precision results.

G. Shanmugasundaram et. al. (2011) proposed an evolutionary model using maturity level of reuse metrics to show the relationship between component based systems, object oriented systems, and service oriented systems using reuse metrics.

Aldeida Aleti and Indika Meedeniya (2011) proposed a Bayesian Heuristic for Component Deployment Optimization (BHCDO) which builds deployment architectures by using a Bayesian learning mechanism. BHCDO efficiently automates the search for component deployment design alternatives and outperforms state of the art optimization methods. BHCDO does not require any parameter tuning to have a good performance as required in other approximate optimization methods.

Mostefai Mohammed Amine and Mohamed Ahmed-Nacer (2011) proposed Knowledge Management System (KMS) implementation in a CBSE-oriented organization that uses short iterations, less resources and budget, continuous integration and intensive customer collaboration to eliminate risks and misconceptions.

Abhikriti Narwal (2012) conducted a survey and interviews with Software developers, Testers, Research Engineers of many major Software Companies to show usage of Complexity metrics in Component-based Software Systems may improve the quality of Software. Complex components are hard to understand and take much time to execute than simple components and are very difficult to maintain.

Amr Rekaby and Ayat Osama (2012) proposed a model containing activities involved in component-based development lifecycle that can decrease the effort of the projects by 40% after few months of application.

Sandeep Srivastava (2012) discussed the relation between software metrics and maintainability which characterize the ease of the maintenance process when applied to a specific product. It is determined that up to what point and in which cases we can rely on software metrics in order to define the maintainability of a software product.

Simrandeep Singh Thapar et. al. (2012) discussed usage of Component Based Software Development (CBSD) approach as a success factor for business that provides benefits like reusability, on-time delivery, high quality, and less cost. The major reason of focus of software organizations to implement quality management in software development is the quality expectations of customers at purchase time from the software.

Anupama Kaur et. al. (2012) proposed a neural network based approach to establish the relationship between different attributes for evaluating reusability. Structural attributes of function oriented software components are shown using software metrics i.e. McCabe's cyclometric complexity measure for complexity measurement, regularity Metric, Halstead Software Science Indicator for Volume indication, Reuse Frequency metric and Coupling Metric.

Ashish Oberoi and Deepti Arora (2014) analyzed CK metric values and used Self Organizing Map for evaluation of CK metric of component quality model. They provided CK metric value to improve the performance using design patterns.

Neha Goyal and Deepali Gupta (2014) proposed a model that uses unsupervised neural network to calculate reusability of a component based software with the help of Rational Rose.

3.0 PROPOSED WOK

Design patterns are considered as separate components for the problem formulation. All Implementation has been done by using MATLAB Version 2013-Neural Network Tool Box. Data Structure, Algorithms, functions and implementation done for doing this work are discussed here. Ashish Oberoi and Deepti Arora (2014) derived quality model using same strategy but no

comparison or evaluation has been done, on the basis of which optimum values was retrieved.

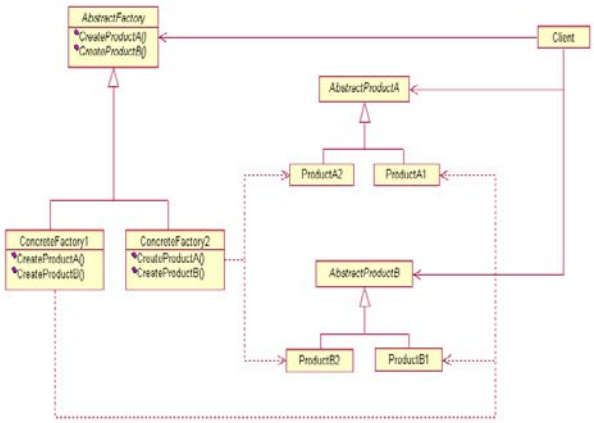
Optimization of Component based Software Engineering model is done through analyzing a series of design patterns which are worldwide accepted as the reuse design terminology for object oriented designing and hence component based designing. In the proposed model, firstly entire design pattern are analyzed obtained from Gamma et. al. (1995) by formulating CK metric analysis. Based on average, standard deviation on metrics values obtained and weight values got through unsupervised neural network Self Organizing Map (SOM); optimized values for components are achieved. Here we are using an unsupervised method because we don't know in advance output values for the corresponding design patterns. MatLab will be used for the implementation purpose. The problem is divided into two phases:

1. Analysis of Software Design Patterns through CK metrics analysis.
2. Implementation through Neural Network using MatLab

3.1. CK Metric Analysis of Software Design Patterns

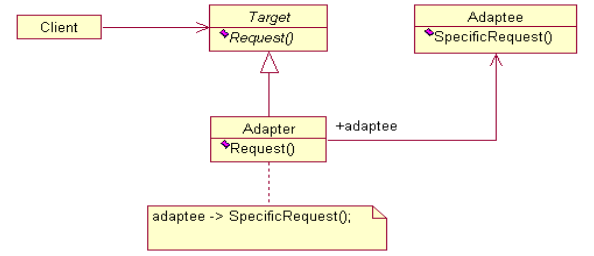
Design patterns are usually used in the software design phase to create abstractions to provide independency in components and to handle future changes and maintaining architectural integrity. In this phase, CK metric values (WMC, DIT, NOC, RFC, CBO) of each component's every class is computed for design patterns. To get the value of metric NC, number of classes in each component is found at run time and added to the final CK metric values. Analysis of CK metric values for design patterns are shown in table 1.

Abstract Factory



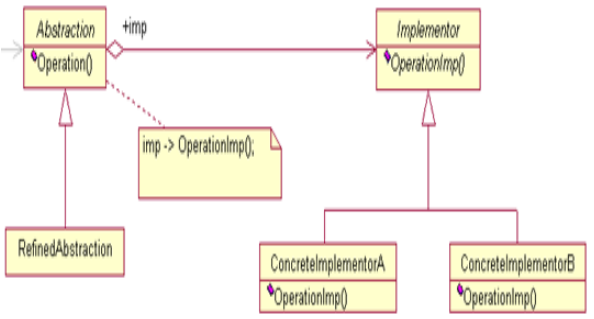
Class/Metrics	NOM	DIT	NOC	CBO	RFC
Abstract Factory	2	0	2	0	2
Concrete Factory	2	1	0	2	2
Concrete Factory	2	1	0	2	2
Abstract Product	0	0	2	0	0
Product A2	0	1	0	0	0
Product A1	0	1	0	0	0
Abstract Product	0	0	2	0	0
Product B2	0	1	0	0	0
Product B1	0	1	0	0	0

Adapter



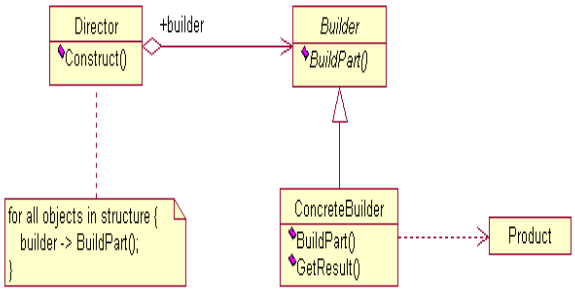
Class/Metrics	NOM	DIT	NOC	CBO	RFC
Target	1	0	1	0	1
Adapter	1	1	0	0	2
Adaptee	1	0	1	0	1

Bridge



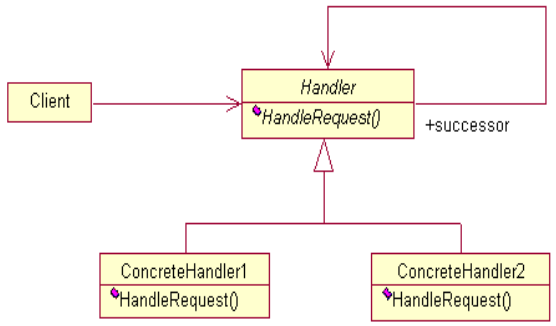
Class/Metrics	NOM	DIT	NOC	CBO	RFC
Abstaction	1	0	1	0	2
Refined Abstraction	0	1	0	0	0
Implementor	1	0	2	0	1
Concrete Implementor A	1	1	0	0	1
Concrete Implementor B	1	1	0	0	1

Builder



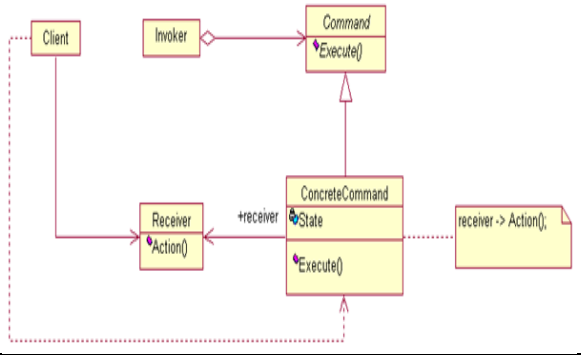
Class/Metrics	NOM	DIT	NOC	CBO	RFC
Director	1	0	0	0	2
Builder	1	0	1	0	1
Concrete Builder	2	1	0	1	2
Product	0	0	0	0	0

Chain of Responsibility



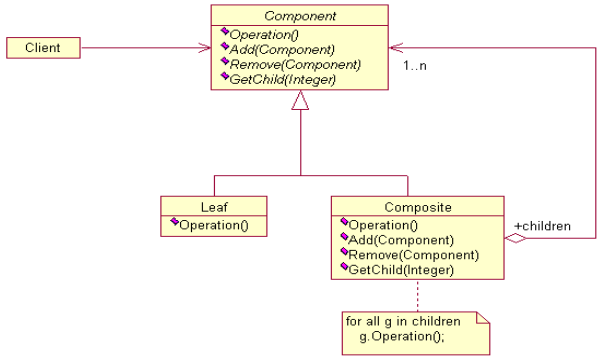
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Handler	1	0	2	0	2
Concrete Handler 1	1	1	0	0	1
Concrete Handler 2	1	1	0	0	1

Command



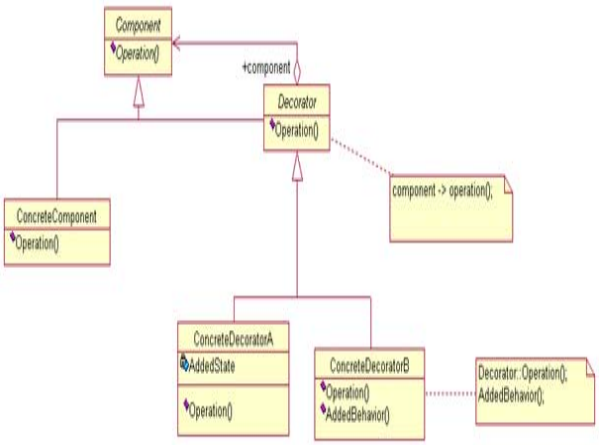
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Invoker	0	0	0	0	0
Command	1	0	1	0	1
Concrete Command	1	1	0	0	2
Receiver	1	0	0	0	1

Composite



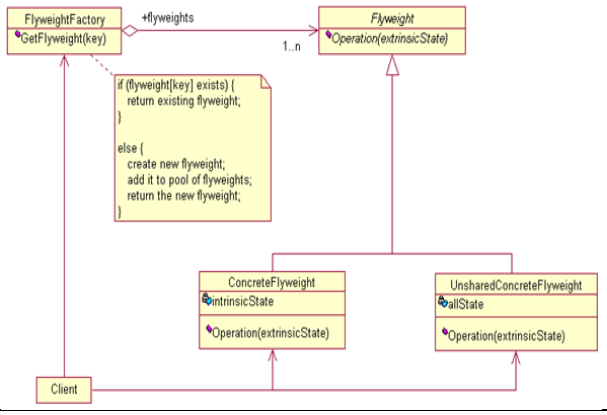
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Component	4	0	2	0	4
Leaf	1	1	0	0	1
Composite	4	1	0	0	5

Decorator



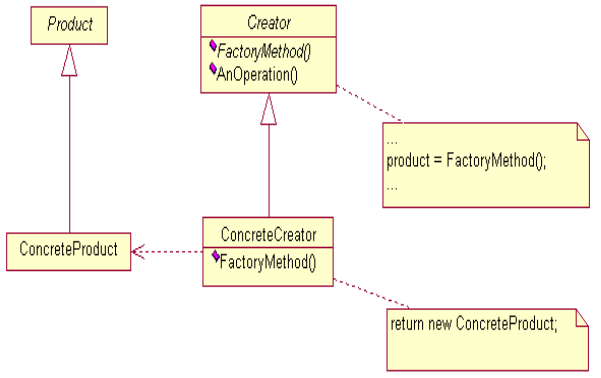
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Component	1	0	2	0	1
Concrete Component	1	1	0	0	1
Decorator	1	1	2	0	2
Concrete Decorator A	1	2	0	0	1
Concrete Decorator B	2	2	0	0	3

Flyweight



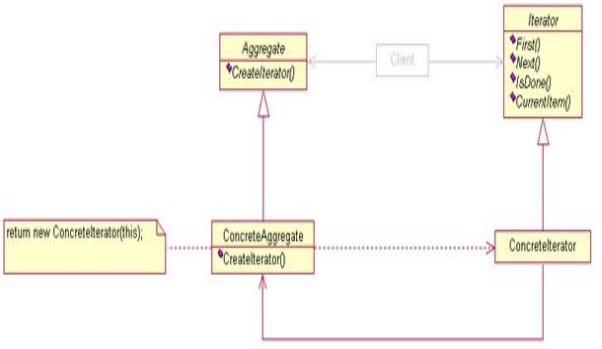
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Fly Weight Factory	1	0	0	0	1
Fly weight	1	0	2	0	1
Concrete Fly weight	1	1	0	0	1
Unshared Concrete Fly weight	1	1	0	0	1

Factory Method



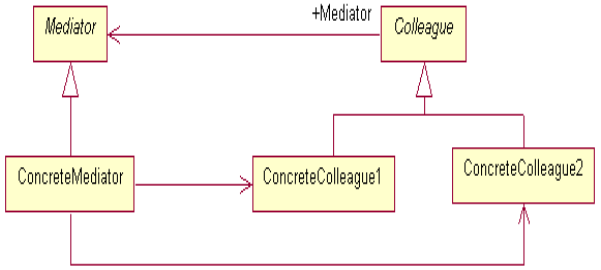
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Product	0	0	2	0	0
Concrete Product	0	1	0	0	0
Creator	2	0	1	0	2
Concrete Creator	1	1	0	1	1

Iterator



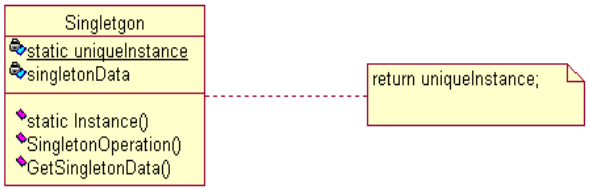
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Aggregate	1	0	1	0	1
Concrete Aggregate	1	1	0	1	2
Iterator	4	0	1	0	4
Concrete Iterator	0	1	0	0	0

Mediator



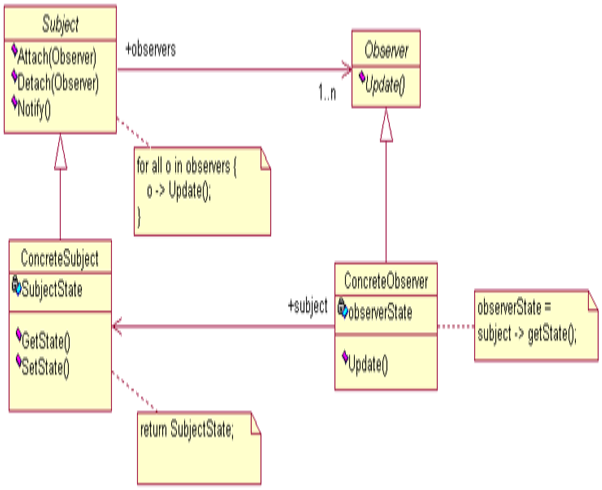
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Mediator	0	0	1	0	0
Colleague	0	0	2	0	0
Concrete Mediator	0	1	0	0	0
Concrete Colleague 1	0	1	0	0	0
Concrete Colleague 2	0	1	0	0	0

Singleton



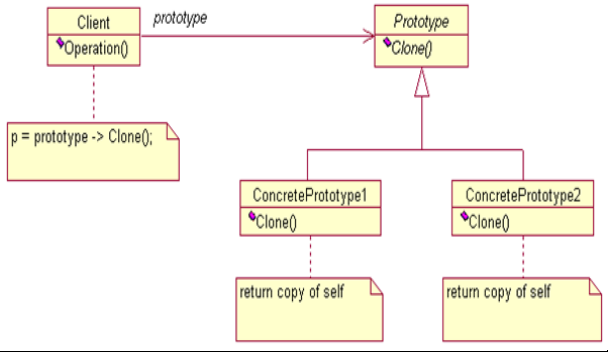
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Singleton	3	0	0	0	3

Observer



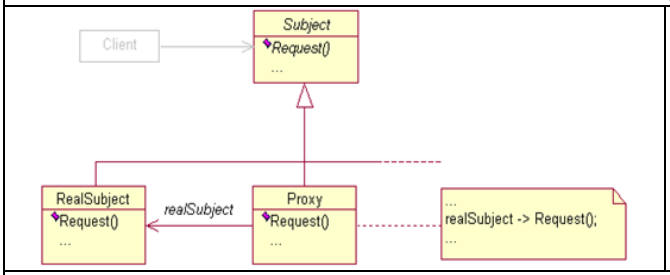
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Subject	3	0	1	0	4
Concrete Subject	2	1	0	0	2
Observer	1	0	1	0	1
Concrete Observer	1	1	0	0	2

Prototype



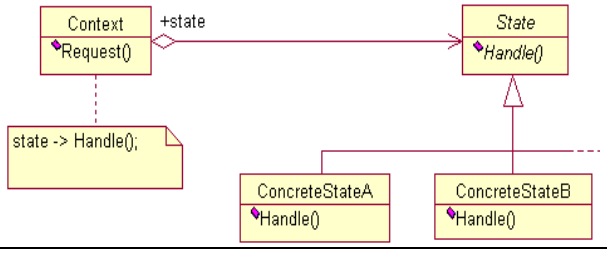
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Prototype	1	0	2	0	1
Concrete Prototype 1	1	1	0	0	1
Concrete Prototype 2	1	1	0	0	1

Proxy



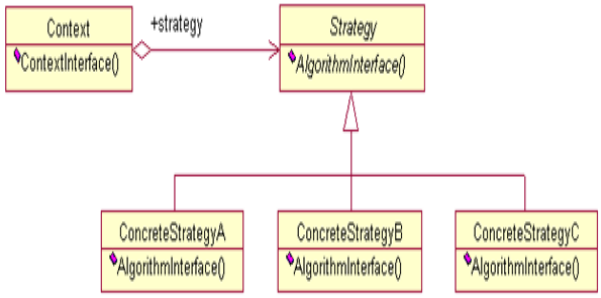
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Subject	1	0	2	0	1
Real Subject	1	1	0	0	1
Proxy	1	1	0	0	2

State



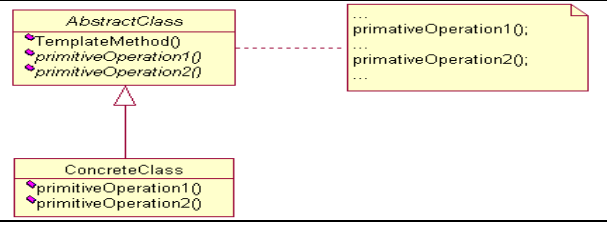
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Context	1	0	0	0	2
State	1	0	2	0	1
Concrete State A	1	1	0	0	1
Concrete State B	1	1	0	0	1

Strategy



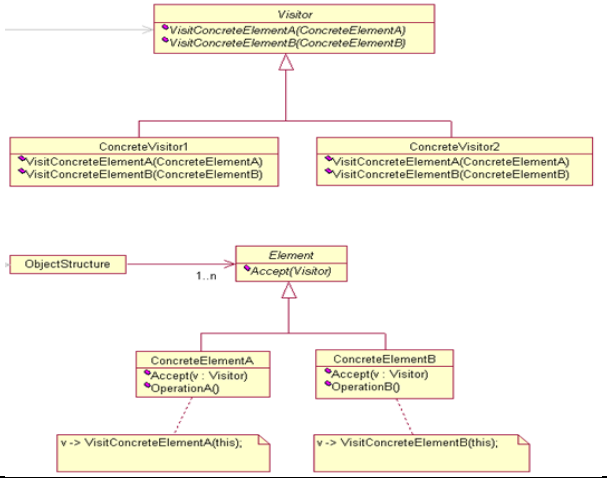
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Context	1	0	0	0	1
Strategy	1	0	3	0	1
Concrete Strategy A	1	1	0	0	1
Concrete Strategy B	1	1	0	0	1
Concrete Strategy C	1	1	0	0	1

Template Method



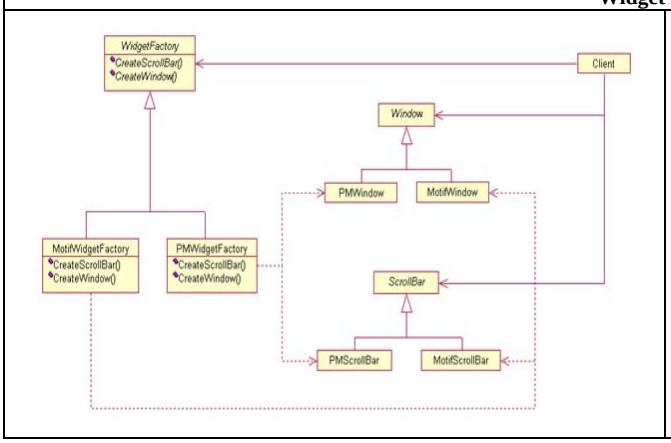
Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Abstract Class	3	0	1	0	5
Concrete Class	2	1	0	0	2

Visitor



Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Visitor	2	0	2	0	2
Concrete Visitor 1	2	1	0	0	2
Concrete Visitor 2	2	1	0	0	2
Object Structure	0	0	0	0	0
Element	1	1	2	0	1
Concrete Element A	2	1	0	0	3
Concrete Element B	2	1	0	0	3

Widget Factory



Class/ Metrics	NOM	DIT	NOC	CBO	RFC
Widget Factory	2	0	2	0	2
Motif Widget Factory	2	1	0	2	2
PM Widget Factory	2	1	0	2	2
Window	0	0	2	0	0
PM Window	0	1	0	0	0
Motif Window	0	1	0	0	0
Scroll Bar	0	0	2	0	0
PM Scroll Bar	0	1	0	0	0
Motif Scroll Bar	0	1	0	0	0

Table 1: Design Pattern with Metric Values

3.2. Implementation using Neural Network

Unsupervised neural network i.e. Self Organizing Map is created using newsom() of Neural Network Toolbox of MatLab. CK metric values of design patterns obtained through first step are feed as input in the created network. The parameters taken are: trainbuwb method as training method, no. of training data is 21x6=126, number of epoch taken to converge are 500. After the training, free parameter value i.e. weight is summed up for each metric. Weighted sum is divided by total number of metrics that returns optimum value for each metric which in turn establishes landmark for reusability of component based model. Also, average and standard deviation values are retrieved to compare with optimum values. For finding average and standard deviation values, mean() and std() of MatLab have been used.

4.0 RESULT ANALYSIS

Graphs showing comparison between Average values, Standard Deviation values and experimental values evaluated through SOM based analysis is shown in figure 1(a-c). CK metric is shown on x-axis and Values w.r.t. CK metric is shown on y-axis.

Variations in average, standard deviation and experimental values are shown in table 2. By getting average of all the variations as shown in table 2, optimum values can be found to improve the reusability of component based software engineering model.

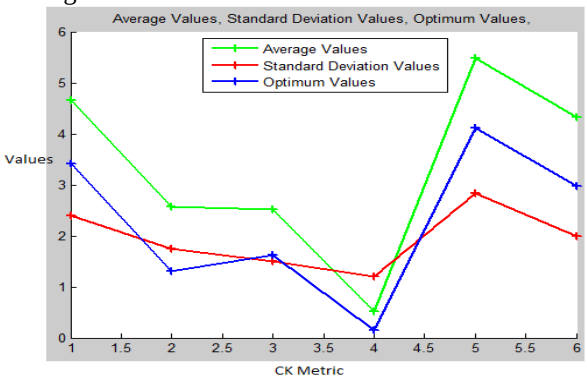


Figure 1(a)

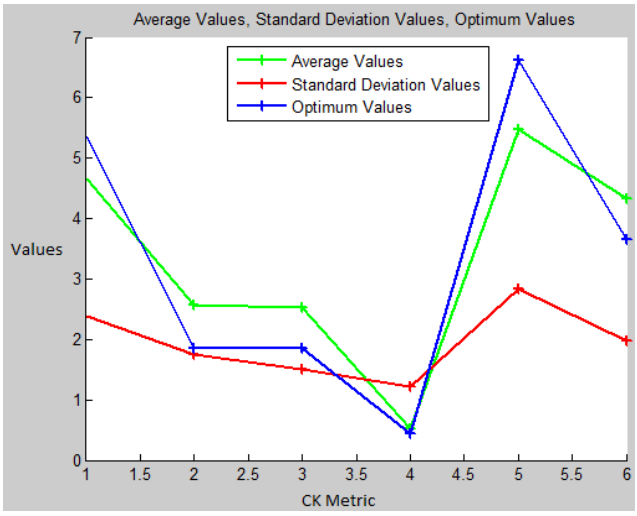


Figure 1(b)

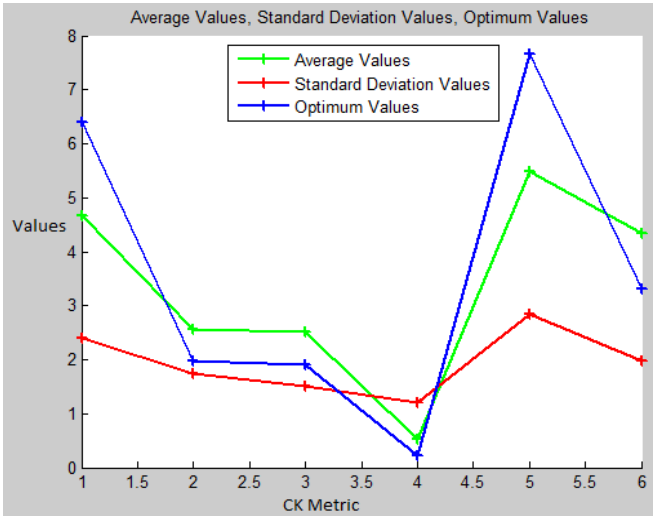


Figure 1(c)

Figure 1: Variation 1 between Average, Standard Deviation, and Optimum value for design patterns.

S. No.	CK Metric	Variation 1	Variation 2	Variation 3	Optimum Values
1	WMC	4	6	7	5.666
2	DIT	2	2	2	2
3	RFC	2	2	2	2
4	NOC	1	1	1	1
5	NC	5	7	8	6.666
6	CBO	3	4	4	3.666

Table 2: Variations between Average, Standard Deviation and Optimum Values

5.0 CONCLUSION

CBSE is a knowledge-intensive activity that helps in producing better quality software systems by playing significant role in achieving programmer's productivity, system flexibility, and overall system quality. A model has been proposed for optimizing CK metric values with respect to the Component Based Software Engineering (CBSE) methodology using design patterns. UML properties have been applied to find out the various metrics from each and every class of various types of design patterns (components). While adding up the metric values for each component, number of classes is calculated at run time and concatenated with the final metrics value. These metric values are feed to un-supervised neural network. Graphs have been plotted with the help of average, standard deviation and optimized values to show the variation. Optimum values achieved on the basis of CK metric variations provide an optimized model for Software Component Engineering model.

6.0 LIMITATIONS & FUTURE SCOPE

For the evaluation of software components, a component quality model is required that reuse not only the functional parts, but also achieve easier and more accurate predictability of the system behavior. The proposed model used for optimization of component based software engineering can give better results if good amount of data is provided with realistic values e.g. historical project values of a software company that have used component based development. If that data is used as an input and results in terms of quality factors like reusability, maintainability, complexity, testability etc. are given as output. A supervised neural network may give better results as compared to unsupervised neural network which is used in the proposed model.

7.0 REFERENCES

[1]. Ashish Oberoi, Deepti Arora, "Quality Model For Analysis And Implentation Of CK Metrics Through Neural Networks", National Conference on Advances in Engineering and Technology, pp. 46-51, March 2014.

[2]. Neha Goyal, Deepali Gupta, "Reusability Calculation of Object Oriented Software Model by Analyzing CK Metric", International Journal of Advanced Research in Computer Engineering & Technology, Vol. 3, Issue 7, pp. 2466-2470, July 2014.

[3]. Rajni Jain, Satma M C, Alka Aroa, Sudeep Marwaha, R C Goyal, "Online Rule Generation Software Process Model", BIJIT - BVICAM's International Journal of Information Technology, Vol. 5, No. 1, pp. 505-511, Jan. – June, 2013.

[4]. MatLab R2013 Neural Network Tool Box Product Help.

[5]. Abhikriti Narwal, "Empirical Evaluation of Metrics for Component Based Software Systems", International Journal of Latest Research in Science and Technology, Vol. 1, Issue 4, pp.373-378, Nov.- Dec. 2012.

[6]. Amr Rekaby, Ayat Osama, "Introducing Integrated Component-Based Development Lifecycle and Model", International Journal of Software Engineering & Applications, Vol. 3, No. 6, pp. 87-99, Nov. 2012.

[7]. Sandeep Srivastava, "Software metrics and Maintainability Relationship with CK Metrics", International Journal of Innovations in Engineering and Technology, Vol. 1 Issue 2, pp. 76-82, Aug. 2012.

[8]. Anupama Kaur, Himanshu Monga, Mnupreet Kaur, Parvinder S. Sandhu, "Identification and Performance Evaluation of Reusable Software Components Based Neural Network", International Journal of Research in Engineering and Technology, Vol. 1, No. 2, pp. 100-104, March 2012.

[9]. Simrandeep Singh Thapar, Paramjeet Singh, Shaveta Rani, "Challenges to the Development of Standard Software Quality Model", International Journal of Computer Applications, Vol. 49, No.10, pp. 1-7, July 2012.

[10]. G. Shanmugasundaram, V. Prasanna Venkatesan, C. Punitha Devi, "Reusability metrics - An Evolution based Study on Object Oriented System, Component based System and Service Oriented System", Journal Of Computing, Volume 3, Issue 9, pp. 30-38, Sept. 2011.

[11]. Aldeida Aleti, Indika Meedeniya, "Component Deployment Optimisation with Bayesian Learning", ACM Journal, pp. 11-20, June 2011.

[12]. Mostefai Mohammed Amine, Mohamed Ahmed-Nacer, "An Agile Methodology For Implementing Knowledge Management Systems : A Case Study In Component-

Based Software Engineering", International Journal of Software Engineering and Its Applications, Vol. 5, No. 4, pp. 159-170, 2011.

[13]. P. C. Jha, Shivani Bali and P. K. Kapur, "Fuzzy Approach for Selecting Optimal COTS Based Software Products Under Consensus Recovery Block Scheme", BIJIT - BVICAM's International Journal of Information Technology, Vol. 3, No. 1, pp. 318-323, Jan. – June 2011.

[14]. Anju Shri, Parvinder S. Sandhu, Vikas Gupta, Sanyam Anand, "Prediction of Reusability of Object Oriented Software Systems using Clustering Approach", World Academy of Science, Engineering and Technology, Vol. 43, pp. 853-856, 2010.

[15]. G. M. Tere and B. T. Jadhav, "Design Patterns for Successful Service Oriented Architecture Implementation", BIJIT - BVICAM's International Journal of Information Technology, Vol. 2, No. 2, pp. 245-249, July – Dec. 2010.

[16]. Sonia Manhas,Rajeev Vashisht,Reeta Bhardwaj, "Framework for Evaluating Reusability of Procedure Oriented System using Metrics based Approach", International Journal of Computer Applications, Vol. 9, No. 10, pp. 14-19, Nov. 2010.

[17]. V. Lakshmi Narasimhan, P. T. Parthasarathy, M. Das, "Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE)", Issues in Informing Science and Information Technology, Vol. 6, pp. 731-740, 2009.

[18]. María A. Reyes, Maryoly Ortega, María Pérez, Anna Grimán Luis E. Mendoza and Kenyer Domínguez, "Toward A Quality Model for CBSE", International Conference on Enterprise Information Systems, pp. 101-106, 2009.

[19]. R. Senthil, D. S. Kushwaha, A. K. Misra, "An Extended Component Model and its evaluation for Reliability & Quality", Journal of Object Technology, Vol. 7, No. 7, pp. 109-129, Sept. 2008.

[20]. Yoonjung Choi, Sungwook Lee, Houp Song, Jingoo Park, SunHee Kim, "Practical S/W Component Quality Evaluation Model", ICACT, pp. 259-264, Feb. 2008.

[21]. Mubarak Mohammad, Vasu Alagar, "A Component-Based Software Engineering Approach for Developing Trustworthy Systems", ACTS Report Series, Feb. 2008.

[22]. Anita Gupta, Reidar Conradi, Forrest Shull, Daniela Cruzes, "Experience Report on the Effect of Software Development Characteristics on Change Distribution", Springer Journal, pp. 158–173, 2008.

[23]. Kung-Kiu Lau, Zheng Wang, "Software Component Models", IEEE Transactions On Software Engineering, Vol. 33, No. 10, pp. 709-724, Oct. 2007.

[24]. Net Objective, "Design Patterns: From Analysis to Implementation", Manuals for design patterns explained: A New perspective for Object Oriented Design, 2007.

[25]. Alexandre Alvaro, Eduardo Santana de Almeida, Silvio Lemos Meira, "A Software Component Quality Model: A Preliminary Evaluation", IEEE Proc. of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, 2006.

[26]. Parvinder S. Sandhu, Hardeep Singh, "Automatic Reusability Appraisal of Software Components using Neuro-fuzzy Approach", International Journal of Information Technology,Vol. 3,No. 3, pp. 209-215, 2006

[27]. Kilsup Lee, Sung Jong Lee, "A Quantitative Software Quality Evaluation Model for the Artifacts of Component Based Development", Proc. of the 6th IEEE International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005.

[28]. Sajjad Mahmood, Richard Lai, Yong Soo Kim, Ji Hong Kim, Seok Cheon Park, Hae Suk Oh, "A survey of component based system quality assurance and assessment", Elsevier Information and Software Technology, Vol. 47, pp. 693-707, 2005.

[29]. Ramanath Subramanyam and 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, pp. 297-310, April 2003.

[30]. Simon Haykin, Neural Networks: A Comprehensive Foundation, Pearson Education, 2002.

[31]. Margaretha W. Price, Donald M. Needham, Steven A. Demurjian, "Producing Reusable Object-Oriented Components: A Domain-and-Organization-Specific Perspective", ACM Proc.of the symposium on Software reusability: putting software reuse in context, pp. 41-50, May 18-20, 2001.

[32]. Tullio Vernazza, Giampiero Granatella, Giancarlo Succi, Luigi Benedicenti, Martin Mintchev, "Defining metrics for software components", Proc. of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. 11, pp. 16-23, July 2000.

[33]. John Grundy, Warwick Mugridge, John Hosking, "Constructing Component-based Software Engineering Environments: Issues and Experiences", Elsevier Journal of Information and Software Technology, Vol. 42, No. 2, pp. 117-128, Jan. 2000.

[34]. Neville I. Churcher, Martin J. Shepperd, "Comments on - A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 21, No. 3, pp. 263-265, March 1995.

[35]. R. Geoff Dromey, "A Model for Software Product Quality", IEEE Transactions on Software Engineering, Vol. 21, No. 2, pp. 146-162, Feb. 1995.

[36]. E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley, 1995

[37]. Shyam R. Chidamber, Chris F. Kemerer, "A metrics suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 476-493, June 1994.

[38]. Sidhu Pravneet, "Quality Metrics Implementation In Component Based Software Engineering Using AI Back Propagation Algorithm Software Component", International Journal of Engineering and Management Sciences, Vol. 3, No. 2, pp. 109-114, 1994.

[39]. James C. Browne, Taejae Lee, John Werth, "Experimental Evaluation of a Reusability-Oriented Parallel Programming Environment", IEEE Transactions on Software Engineering, Vol. 16. No. 2, pp. 111-120, Feb. 1990.