2012 2nd International Conference on Information Communication and Management (ICICM 2012)
IPCSIT vol. 55 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V55.4

Measuring Complexity of Component Based System

Using Weighted Assignment Technique

Rajender Singh Chillar[1+], Priyanka Ahlawat[2] and Usha Kumari[3]

[1] Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak (Haryana)
[2] Department of Computer Engineering, National Institute of Technology, kurukshetra (Haryana)
[3]A.I.J.H.M (PG) College, Rohtak (Haryana), chhillarusha01@gmail.com

Abstract. Component based software engineering advocates acquisition, variation and integration of reusable software components to rapidly develop complex systems with minimum effort and cost. Components are essential part of component based software development. Researchers are striving hard to identify metrics that help in estimating complexity of component based system. Interactions /interfaces among components play a major role in contributing complexity to a component based system. In this paper, we have proposed two metrics for measuring complexity of interface and interface dependency of component based system. These metrics are based on different constituents of an interface like interface methods and instance variables with different weights assigned to them. These interfaces can be represented by adjacency matrix used in graph theory. Strength of proposed metrics is computed using weighted assignment technique. Empirical evaluation is done using a case study. The metrics are also validated on Java Beans. Result of present study is quite encouraging and may further help in estimating complexity of components.

Keywords: Component based software engineering, Component, Complexity of interface, Interaction, Weighted assignment technique.

1. Introduction

Component based software development is process of assembling various software components in an application such that they interact to satisfy a predefined functionality [1]. Most of research is devoted to improving the methods and approaches of development process. A very little work has been devoted for evaluation of components using metrics [2]. Software metrics are broadly classified as product and process metrics where former aims at measuring the attributes of product and latter measures the development or maintenance process and their environment [2, 3]. Software metrics describe a wide range of activities concerned with measurement in software engineering [3]. Traditional metrics include lines of code, cyclomatic complexity, Halsted metric, function point analysis, bugs or faults per line of code. These metrics are based on source code of component which is usually unavailable in component based system.

The rest of paper is organized as follows. Section 2 discusses the related work on component oriented metrics. Section 3 presents the Proposed Metrics. Section 4 empirically evaluates the Proposed Metrics. Section 5 deals with validation on Java Beans followed by conclusion in section 6.

2. Component Oriented Complexity Metrics

Various metrics have been developed by researchers for improving the quality of software components. Cho and Kim proposed metrics for quality attributes such as complexity, customizability and reusability. They classified these metrics as design time metrics and implementation metrics [4]. In another paper, Boxall

+ Corresponding author. *E-mail address*: chhillar02@gmail.com

and Araban proposed a set of interface metrics for reusability analysis of components [5]. Chidamber and Kemerer proposed a suite of six metrics namely number of children, depth of inheritance tree, weighted methods per class, coupling between objects, lack of cohesion in methods, response for class [7].

3. Proposed Complexity Metrics For Component Based System

A component based system is obtained as a result of composition of some other component through well defined interfaces. These create interaction that promotes dependencies among components [6]. In a component based system, complexity depends not only on component but also on its interaction process. A component specification includes component interface in terms of services provided and used, interactions between components, the way they interact in form of context dependency and constraints [8]. Interfaces of a component act as an access point to other component. Interfaces/ interaction can either be incoming or outgoing [6]. These are primary source for understanding, use and implementation for component. Thus complexity of interface has significant contribution to overall complexity of component. Content of interaction play a significant role in deciding overall interaction complexity [8].

A component can either be primitive or compound. A compound component is collection of many other components. The complexity of interface of a component can be measured in terms of size of interface (component own functionality), interface coupling with internal subcomponents and interface coupling with outer components. Interface coupling is defined in terms of the number of methods and instance variables invoked by component from other components .Weighted values are assigned to constituents of interface coupling (methods and instance variables).Interface methods are classified according to data type of arguments and return values. Basic assumption in computing data type of instance variables, arguments and return type (interface methods) is : Primitive data types such as integer will be simple, Structured data type such as string, array, list, and vector will be medium ,Complex data types includes class type, user defined components, pointers, references and others .If arguments are different for same interface method then higher data type will be taken. Simple <Medium<Complex (order of priority). Similarly we compute data type of return types. Interface methods on basis of data type of arguments and return types can be classified as simple(S), medium (M) and complex(C) by using Table1

Return types /arguments	**No**	**primitive**	**structured**	**Complex**
No	S	S	M	M
primitive	S	S	M	M
structured	S	M	C	C
complex	M	M	C	C

Table -1: Categorization of type of interface methods

From the study of [1,2,5,6,9,10,11,12] ,We propose metrics for evaluating complexity of component based system namely complexity of interface and interface dependency. For the proposed metrics, we consider interfaces of components. A component interfaces with other components by invoking methods and instance variables. Strength of metrics is computed using the number of methods and instance variables. A directed graph and adjacency matrix is used to represent interfaces in a component based system. In a directed graph, an edge from $component_i$ to $component_j$ represents an interface (external). Interface Coupling is defined in terms of methods and instance variables invoked by $component_i$ from $component_j$. Each edge(value) is represented by m_{ji},v_{ji}.where m_{ji} represents number of methods invoked and v_{ji} represents number of instance variables used by $component_i$ from component $_j$ [11].Thus adjacency matrix M may be used to represent the directed graph where

M [i, j] =0 (if no edge from $Component_i$ to $component_j$, if $component_i$ not directly connected with $component_j$).

= m_{ji},v_{ji} (if component$_i$ is directly connected with component$_j$ and m_{ji} is number of methods invoked by component$_i$ from component$_j$ and v_{ji} number of instance variables invoked by component$_i$ from component$_j$. From directed graph and adjancy matrix, complexity of an interface component can be computed as:

Complexity of interface $_{component}$ (C.I) =CSI+ICC (internal) +ICC (external) **(1)**

CSI= Complexity of size of interface, ICC(External) =Interface coupling complexity (External to Component)) ,ICC(Internal) =Interface coupling complexity (Internal to Component))

Component interface size is total number of instance variables and interface methods contained by component.=[(w1∑m(simple)+w2∑m(medium)+w3∑m(complex))+(w4∑v(simple)+w5∑v(medium)+w6∑v (complex))] where w1to w6 are weight values

ICC (Internal) = [(∑ w7 (m/v invoked (simple) + ∑w8 (m/v invoked (medium) + ∑w9 (m/v invoked (complex)]). where w7 to w9 are weight values of methods and instance variables invoked within subcomponents in a component i.e. internal to a component

ICC (External) = [w10 (∑ (m/v invoked (simple) +w11 ∑ (m/v invoked (medium) + w12∑ (m/v invoked (complex)]) where w10 to w12 are weight values of methods and instance variables invoked by a component in from other components in a component based system i.e. external to component.

Total Complexity of component based system = $\sum_{i=1}^{n} C$ I ,n=total number of components in a system **(2)**

Average Complexity of component based system = $\sum_{i=1}^{n} C$ I / n, n=total number of components in a system **(3)**

A component based system is composed of various components assembled together .One component depends on other component for its functionality. Thus dependency of a component can be defined as the functionality obtained from other components apart from its own functionality. Dependencies can be shown using a graph and its corresponding adjacency matrix.To compute interface dependency metric, we use a directed graph and adjacency matrix where

M [i,j]=0 (if there is no edge from component$_i$ to component$_j$ i.e. component$_i$ does not depend on component$_j$ for its functionality).

=m_{ji},v_{ji} (if there is an edge from component$_i$ to component$_j$ i.e. component$_i$ obtains functionality from component$_j$ and m_{ji} is number of methods invoked by component$_i$ from component$_j$ and v_{ji} number of instance variables invoked by component$_i$ from component$_j$.

Interface Dependency Metric (IDM$_{component}$) =functionality obtained from other components/ functionality ($_{component}$) **(4)**

Functionality is defined in terms of methods and instance variables .A component obtain functionality by invoking methods and instance variable from other components in a component based system.

***Functionality$_{component}$=w*13∑m(simple)+w14∑m(medium)+w15∑m(complex))+(w16∑v(simple)+w17 ∑v(medium)+w18∑v(complex))] where w13 to w18 are weight values** **(5)**

Functionality obtained from other components = [(∑ w19 (m/v invoked (simple) + ∑w20 (m/v invoked (medium) + ∑w21 (m/v invoked (complex)]). where w19 to w21 are weight values of methods and instance variables invoked **(6)**

Total Interface dependency metric of component based system = ∑IDM from i to n (n=total number of components in a system) **(7)**

Average Interface dependency metric of component based system = ∑IDM (from i to n) /n (n=total number of components in a system) **(8)**

4. Empirical Evaluation Of Proposed Metric Using Weighted Assignment Technique

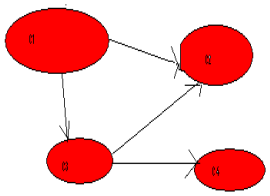


Fig1 Weighted edge graph

Adjancey matrix	C1	C2	C3	C4
C1	0	2,3	0	2,2
C2	0	0	0	0
C3	0	0	0	0
C4	0	2,4	2,3	0

Table 2 Assumed data in adjacency matrix from weighted edge graph

Suppose we have a component based system that consists of four components and four interfaces among them. A node represents component and edge represents interfaces among components. There are four edges hence four interface coupling. Complexity of interface can be computed by number of methods and instance variables invoked during an interface coupling. In this case study, we assume all components have simple methods and variables. In an adjacency matrix, M [Ci, Cj] represents an interface in terms of methods and instance variables invoked by $component_i$ from $component_j$. The relative data, weight values are assumed and given in Table 3 and Table 4.

Table 3 &4: Weight values assigned to interface methods and Complexity of size of interface (component own methods and variables) For instance variables, we assign the following weights: simple-.02, medium-.04, and complex-.06

Data type - Number	Simple	Medium	Complex
0-2	.02	.04	.06
3-5	.04	.06	.08
6-8	.06	.08	.10
9-11	.08	.10	.12
>=12	.10	.12	.14

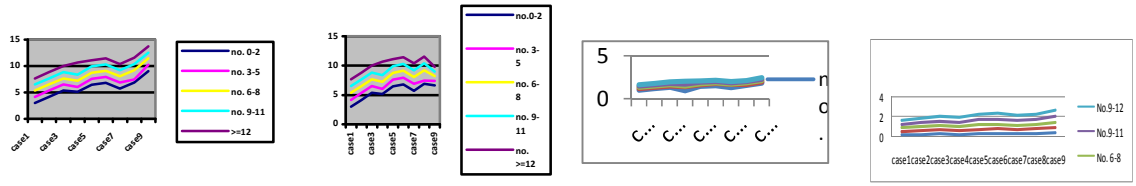
Name of component	Number of methods. number of variables(both are simple and number of arguments passed or return types are in range 0-2 and simple)
C1	5,6=5*.02+6*.02=.22
C2	2,3=.10
C3	4,5=.18
C4	2,3=.10

Table 5: Computed values of complexity of interface metric and interface dependency metric from case study

Case1 :instance variables(simple)and methods(simple)						
Number of Parameters(arguments and return values)	Component c1(complexity of interface, interface dependency metric)	Component c2(complexity of interface ,interface dependency metric)	Component c3(complexity of interface, interface dependency metric)	Component c4(complexity of interface ,interface dependency metric)	Total complexity of component based =∑complexity of CI(i) where i=1 to n., total interface dependency metric	Average complexity of component based system, average interface dependency metric
0-2	.18+.22,.818	0+.10,0	0.18+0,0	.22+.10,2.2	1.00,3.018	.25,.7545
3-5	.26+.22,1.18	0+.10,0	0.18+0,0	.30+.10,3	1.16,4.18	.29,1.045
6-8	.34+.22,1.545	0+.10,0	0.18+0,0	.38+10,3.8	1.32,5.345	.33,1.336

9-11	.42+.22,1.902	0+.10,0	0.18+0,0	.46+.10,4.6	1.48,6.509	.37,163
>=12	.5+.22,2.272	0+.10,0	0.18+0,0	.54+.105.4	1.64,7.672	.41,1.918
Case2:method (medium) ,instance variables(simple)						
0-2	.26+.22,1.18	0+.10,0	0.18+0,0	.30+.10,3	1.16,4.18	.29,1.05
3-5	.34+.22,1.545	0+.10,0	0.18+0,0	.38+.10,3.8	1.32,5.345	.33,1.33
6-8	.42+.22,1.909	0+.10,0	0.18+0,0	.46+.10,4.6	1.48,6.509	.37,1.627
9-11	.5+.22,2.272	0+.10,0	0.18+0,0	.54+.10,5.4	1.64,7.672	.41,1.918
>=12	.58+.22,2.636	0+.10,0	0.18+0,0	.62+.10,6.2	1.80,8.836	.45,2.209
Case3:method (complex), instance variable(simple)						
0-2	.34+.22,1.545	0+.10,0	0.18+0,0	.38+.10,3.8	1.32,5.345	.33,1.336
3-5	.42+.22,1.909	0+.10,0	0.18+0,0	.46+.10,4.6	1.48,6.509	.37,1.627
6-8	.5+.22,2.27	0+.10,0	0.18+0,0	.54+.10,5.4	1.64,7.672	.41,1.917
9-11	.58+.22,2.636	0+.10,0	0.18+0,0	.62+.10,6.32	1.80,8.836	.45,2.209
>=12	.66+.22,3	0+.10,0	0.18+0,0	.70+.10,7	1.96,10	.49,2.5
Case4 method (simple) and instance variable(medium)						
0-2	.16+.22,1.727	0+.10,0	0.18+0,0	.34+.10,4.4	.8,6.12	.21.53
3-5	.36+.22,1.63	0+.10,0	0.18+0,0	.44+.10,4.4	1.40,6.03	.35,1.5075
6-8	.44+.22,2	0+.10,0	0.18+0,0	.52+.10,5.2	1.46,7.2	.365,1.8
9-11	.52+.22,2.36	0+.10,0	0.18+0,0	.60+.10,6	1.72,8.36	.43,2.09
>=12	.68+.22,3.090	0+.10,0	0.18+0,0	.76+.10,7.6	2.04,10.69	.51,2.522
Case 5 : method (medium) and instance variables(medium)						
0-2	.36+.22,1.636	0+.10,0	0.18+0,0	.48+.10,4.8	1.44,6.436	.36,1.6075
3-5	.44+.22,2	0+.10,0	0.18+0,,0	.56+.10,5.6	1.60,7.6	.4,1.9
6-8	.52+.22,2.36	0+.10,0	0.18+0,0	.64+.10,6.4	1.76,8.76	.44,2.19
9-11	.60+.22,2.72	0+.10,0	0.18+0,0	.72+.10,7.2	1.92,9.92	.48,2.48
>=12	.68+.22,3.090	0+.10,0	0.18+0,0	.80+.10,8	2.08,11.09	.52,2.775
Case 6: method (complex) and instance variable(medium)						
0-2	.44+.22,2	0+.10,0	0.18+0,0	.48+.10,4.8	1.52,6.8	.38,1.7
3-5	.52+.22,2.36	0+.10,0	0.18+0,0	.56+.10,5.6	1.78,7.96	.445,1.99
6-8	.60+.22,2.72	0+.10,0	0.18+0,0	.64+.10,6.4	1.87,9.12	.4675,2.28
9-11	.68+.22,3.090	0+.10,0	0.18+0,0	.72+.10,7.2	2.00,10.29	.5,2.57
>=12	.76+.22,3.45	0+.10,0	0.18+0,0	.80+.10,8	2.16,11.45	.54,2.86
Case 7 method (simple)and instance variable(complex)						
0-2	.38+.22,1.727	0+.10,0	0.18+0,0	.40+.10,4	1.28,5.727	.32,1.43
3-5	.46+.22,2.09	0+.10,0	0.18+0,0	.48+.10,4.8	1.64,6.89	.41,1.7225
6-8	.54+.22,2.45	0+.10,0	0.18+0,0	.56+.10,5.6	1.70,8.05	.425,2
9-11	.62+.22,2.818	0+.10,0	0.18+0,0	.64+.10,6.4	1.86,9.218	.465,2.3045
>=12	.70+.22,3.18	0+.10,0	0.18+0,0	.72+.10,7.2	2.02,10.38	.505,2.595
Case 8:method (medium) , instance variable (complex)						
0-2	.46+.22,2.090	0+.10,0	0.18+0,0	.48+.10,4.8	1.54,6.89	.385,1.7225
3-5	.54+.22,2.45	0+.10,0	0.18+0,0	.50+.10,5	1.64,7.45	.41,1.8625

6-8	.62+.22,2.818	0+.10,0	0.18+0,0	.64+.10,6.4	1.86,9.218	.465,2.3045
9-11	.70+.22,3.18	0+.10,0	0.18+0,0	.72+.10,7.2	2.02,10.38	.505,2.595
>=12	.78+.22,3.545	0+.10,0	0.18+0,0	.80+.10,8	2.12,11.545	.53,2.886
Case 9: method (complex) and instance variable (complex)						
0-2	.54+.22,2.45	0+.10,0	0.18+0,0	.66+.10,6.6	1.80,9.05	.45,2.2625
3-5	.62+.22,2.818	0+.10,0	0.18+0,0	.74+.10,7.4	1.96,10.218	.49,2.5545
6-8	.71+.22,3.227	0+.10,0	0.18+0,0	.82+.10,8.2	2.13,11.427	.5325,2.8
9-11	.78+.22,3.545	0+.10,0	0.18+0,0	.90+.10,9	2.28,12.545	.57,3.136
>=12	.86+.22,3.909	0+.10,0	0.18+0,0	.98+.10,9.8	2.44,13.709	.61,3.427



(a) (b) (c) (d)

Fig 2: (a) Total complexity of interface of component based system vs. number of methods and instance variables and (b)Average complexity of interface of component based system vs. number of methods and instance variables(c)Total interface dependency metric of component based system vs. number of methods and instance variables and (d) Average interface dependency metric of component based system vs. number of methods and instance variables (case study).

5. Validation on Java Beans

Java beans1	Java beans2	Interface coupling external complexity of java beans1,
CFSession Complexity of own methods and variables=2.32 CFSession: complexity of interface (2.48+2.32) =4.80(18 interactions /interfaces) Interface dependency metric (CFSession)= .052+.017+.5568+.0345+.0431+ .017+.043+.0345+.0259+.0172+.129+.0259+.0345+.1206+.3275 +.0259+.0342=1.809/2.32=0.7797	CFCabinetlist	.12
	Integer	.04
	Faces context	.24
	http servlet request	.08
	CF call broker	.10
	String	.04
	System	.02
	CFUtility	.10
	CF Call status	.02
	CF parser	.08
	CFinput xmL	.06
	Calendar	.04
	CFcabinet	.30
	CF preferences	.06
	CFconst	.08
	CFcall status	.28
	CFXMLTAGS	.76
	simple data format	.06
CFCabinetmgmtcomplexity=.60+1.74=2.34 CFCabinetmgmt complexity of interface.= (.08+2.3)=2.38(one interface/interaction), Interface dependency metric (Cabinetmgm)=.08/2.34=.0342	CFSession	.08

Table 6: Validation of Complexity of interface and Interface Dependency Metric on Java Beans

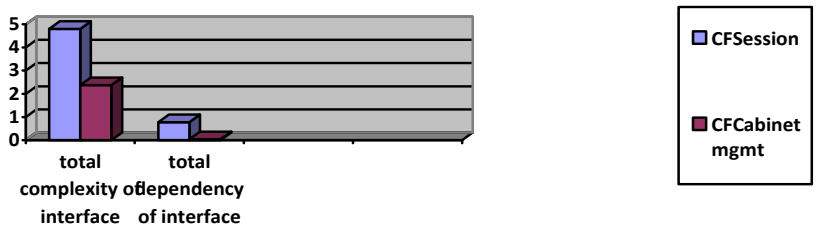


Fig 3: Total complexity of interface vs. total interface dependency metric

We studied a project having Java Beans and computed complexity of interface and interface dependency. Java beans are taken of different sizes, having different functionalities. In the Table 6, Java Bean1 interfaces with Java Bean2 in terms of methods invoked and instance variables used. Strength of interface dependency metric and complexity of interface is computed using weighted assignment technique. The results show that higher dependency among components increases complexity because of more coupling. From figure 5 ,we have noticed that there is positive relation between complexity of interface metric and interface dependency metric .From figure 2 and 3,it is clear that complexity of interface and dependency of interface increase with increase in parameters involved .

6. Conclusion

This paper discusses various complexity metrics for component based system. Two metrics are proposed for measuring complexity of interface and interface dependency in a component based system .Graph theoretic notions are used in arriving at the results. Metrics are applied on Java Beans for empirical evaluation. Strength of proposed metrics is computed using weighted assignment technique. The results show that complexity and dependency increases with increase in number of invoked methods and instance variables during an interface in a component based system.

7. References

[1] N.S.Gill, Balkishan. Dependency and interaction oriented Complexity metrics of component based systems. ACM Sigsoft Software Engineering notes, vol 33, issue 2, pp1-5.

[2] Jianguo Chen, Hui Wang, Yongxia Zhon, Stefan d.Bruda. Complexity metrics for Component based Software System. International Journal of Digital Content Technology and its Applications, Volume 5, Number 3, March 2011.

[3] Jianguo Chen, Wai k Yeap, Stefan D.Bruda. A Review of Component Coupling metrics for component based development, World Congress on Software Engineering, 2009.

[4] E.S.Cho, M.S.Kim & S.D.Kim. Component Metrics to Measure Component Quality. Proceedings of Eight Asia Pacific Software Engineering Conference, Macau, pp.419-426, 2001.

[5] Marcus, Boxall, Saeed Araban. Interface Metrics for Reusability Analysis of components. Proceedings of 2004 Australian Software Engineering Conference (ASWEC'04), Melbourne, Australia, pp: 40-46.

[6] Usha Kumari, Shuchita Upadhyaya. An Interface Complexity Measure for Component based Software Systems. International Journal of Computer Applications (0975-8887), Volume 36 no1, Dec 2011.

[7] Chidamber, Kemerer. A metrics suite for object oriented design. IEEE transactions on Software Engineering, Vol .20, pp 476-493.

[8] Sajjad Mahmood, Richard Lai, .Measuring the Complexity of a UML Component Specification. Proceedings of fifth international Conference on Quality Software (QSIC'05), pp: 150-157.

[9] Kuljit kaur Chahal, Hardeep Singh. A metrics based approach to evaluate design of software components. IEEE International Conference on Global Software Engineering 2008.

[10] Arun sharma,Rajesh Kumar & P.S.Grover. Empirical evaluation and critical review of complexity metrics for software components, Proceedings of 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, Corfu island Greece, pp24-29, 2007.

[11] O.P.Rotaru, Dobre & Petrescu, Reusability metrics for Software Components , Proceeding of 3rd ACSI/IEEE International Conference on Computer Systems and Applications.(AICCSA-05),Cairo,Egypt,pp 24-29.

[12] Ramanpreet Kaur. Evaluation of Software Complexity using Weighted Assignment Technique for Component based System. M.tech Thesis.

[13] Sandeep Khimta, Parvinder S.Sandhu and Amanpreet Brar. A Complexity measure for java bean based software components, World Academy of Science, Engineering and Technology, Volume 42, 2008.

[14] E.J.Weyuker. Evaluating Software Complexity Measures. IEEE Transactions on Software Engineering, Vol. 14, Issue 9, pp 1357-1365.

[15] V.lakshmi Narasimhan,B.Hundradjaya, Detailed theoretical considerations for Suite of Metrics for integration of software components, T.Sobh & K.Ellithy(eds),Advance in Systems, Computing Science & Software Engineering ,257-264,2006,Springer.

[16] A.sharma, R, Kumar, P.S.Grover. Empirical evaluation & validation of interface complexity metrics for student's components. International Journal of Software Engineering & Knowledge Engineering, vol 18, issue 7, pp 919-931.

[17] Iqbaldeep Kaur, Parvinder S. Sandhu, Hardeep Singh and Vandana Saini .Analytical study of Component based Software Engineering. World Academy of Science, Engineering and Technology, Volume 50, pp: 437-442, 2009.

[18] L.Kharb, R.Singh. Complexity metrics for Component oriented Software Systems. ACM SIGSOFT Software Engineering Notes, Vol 33, Issue 2, pp 1-3.

[19] L.Yu, A.Mishra & R.Ramaswamy. Component co –evolution and Component Dependency: speculations and verifications. IET softw. Vol -4, issue 4, pp 252-267, 2010.

[20] S.Mishra. An object oriented complexity metric based on cognitive weights. Proceedings of 6th IEEE international conference on cognitive informatics (ICCI'07).

[21] Tullio Vernazza, Giampiero Granatella, Giancarlo Socci, Luigi Beneficent and Martin Mintchev. Defining metrics for Software components.

[22] Washizaki, Hiraguchi and Yoshiaki fukazana. A metrics suite for measuring quality characteristics of java beans components.PROFES 2008, LNCS 5089, pp 45-60, 2008.

[23] A. Sharma, P.S.Grover & R.kumar. Dependency analysis for component based software systems. SIGSOFT Software Engineering Notes, Vol 34, July 2009.

[24] G.Gui, D.S.Paul. Ranking reusability of software components using coupling metrics .Journal of Systems and Software, Vol. 80, Issue .9, pp: 1450-1459.

Dr. Rajender Singh Chhillar is Professor & Head in the Department of Computer Science and Applications (DCSA), Maharshi Dayanand University (MDU), Rohtak, Haryana, India. He also headed DCSA earlier (2003-2006), the Department of Engineering and Technology (UIET) from April, 2006 to August 2007. Also acted as Director (2003-2010) Computer Centre, MDU and member, monitoring committee of campus wide Networking, M. D. University, Rohtak. He obtained his Master's Degree in Computer Science from Kurukshetra University, Kurukshetra and Doctorate in Computer Science from Maharshi Dayanand University, Rohtak. His research interest includes Software Engineering focusing on Software Metrics, Software Testing, Data Warehousing and Data Mining. He has more than 90 publications in International and National journals. Dr. Singh authored two books; Software Engineering: Metrics, Testing and Faults, Excel Books publisher, New Delhi; and Application of Information Technology to Business, Ramesh Books House, Jaipur. He is a member of various academic bodies like Academic Council, MDU and Computer Society of India (CSI) and editor of various International and National journals.

Er. Priyanka Ahlawat received B.E and M.Tech degree in Computer Science and Engineering from Maharashi Dayanand University Rohtak and Guru Jmbeswar University of Science and Technology Hisar in 2003 and 2006 respectively. She joined department of Computer Engineering in National institute of Technology ,kurushetra(Iindia) in 2008.Curently she is Assistant Professor in Department of Computer Engineerin ,NIT,kurushetra(india).Her research areas include information processing and security component based metrics.

Dr. Usha Chhillar obtained her Master's Degree in Computer Science from Maharshi Dayanand University, Rohtak, M.Phil(Computer Science) from Chaudhary Devi Lal University (CDLU), Sirsa. and Ph.D in Computer Science from Kurukshetra University, Kurukshetra, Haryana, India. She is working as Sr. Lecturer and Head in Department of Computer Science, AIJHM PG College, Rohtak. Her research interest includes Software Engineering focusing on Object Oriented and Component-based metrics.