Dependency Oriented Complexity Metrics to Detect Rippling Related Design Defects

K. Narendar Reddy [1] and A. Ananda Rao [2]
[1] Dept. of CSE, CVR College of Engg., JNTU, Hyderabad, AP, India
[2] JNTU College of Engg., JNTU, Anantapur, AP, India
[1] knreddy_cvr@yahoo.com, [2] akepogu@yahoo.co.in

Abstract

Even though object oriented software development has gained popularity due to its inherent features, it also throws challenges in early detection of defects during design phase. Detection of design defects helps in performing appropriate refactorings in improving the quality of design. Literature indicates that active research is going on in detecting design defects using metrics. The present paper introduces a set of metrics for detecting defects in object oriented designs caused by the presence of shotgun surgery and divergent change bad smells. These metrics are, dependency oriented complexity metric for structure (DOCMS(R)), dependency oriented complexity metric for an artifact causing ripples (DOCMA(CR)), and dependency oriented complexity metric for an artifact affected by ripples (DOCMA(AR)). The proposed metrics have been computed for four cases. These metrics are used successfully in detecting design defects and complexity. In the present study DOCMA(CR) metric value indicated the presence of shotgun surgery bad smell, whereas DOCMA(AR) metric value indicated the presence of divergent change bad smell. DOCMS(R) metric value indicated the increase in complexity of structure (architecture) when the design defects are present. Detecting bad smells helps in performing appropriate refactorings to make the software maintainable and to improve the quality of software.

Keywords: Dcoupling, Dependency Oriented Complexity metric for Structure (DOCMS(R)), Dependency Oriented Complexity Metric for an Artifact Causing Ripples (DOCMA(CR)), Dependency Oriented Complexity Metric for an Artifact Affected by Ripples (DOCMA(AR)).

1. Introduction

Object oriented systems are subject to frequent modifications during software development (iterative or agile software development) and software evolution. As the software is enhanced, modified, and adapted to new requirements the software becomes more complex and deviates from its original design, in turn lowering the quality of software due to design defects. Presence of defects in the design leads to complex system, exhibits faulty behavior, and in turn leads to poor maintainability [1]. It is necessary to detect and correct design defects to make software maintainable. One of the ways to make object oriented software systems maintainable is refactoring. In the context of software under evolution, refactoring is used to improve the software quality. The improvement in the software quality is, in terms of, complexity, maintainability, extensibility, modularity, reusability, and efficiency [2]. Effective refactoring depends on proper quantitative method which is used to detect design defects (which indicate locations / situations for refactoring). In this paper three metrics are proposed to detect the design defects quantitatively. Quantitative detection avoids bias and is amicable for automation. Some strategies for design defects detection are proposed by [3]. These strategies are based on bad smells and metrics. But there is a scope for new metrics which considers ripple effects. Metrics to detect design defects which consider design change propagation probabilities between various artifacts that are connected through intermediate artifacts have been proposed in this paper. This type of metrics which are used to detect design defects caused by the presence of shotgun surgery and divergent change bad smells are not addressed in the literature.

The organization of the paper is as follows. In Section 2 Dcoupling definition is given and proposed metrics are given in Section 3. The analysis results of four case studies are presented in section 4. Section 5 presents the related work on design defects detection methods and metrics. Conclusions have been placed in section 6.

2. Dcoupling Definition and Estimation

The degree of coupling (Dcoupling) of link is the indicator of the amount of dependency that exists between two related artifacts represented by the URA. The design is represented as a unified representation of artifacts (URA) graph [11]. The artifacts in URA graph, map to physical entities in different ways like classes, sets of classes, subsystems etc. The value of Dcoupling has the range [0,1]. Therefore, a change is propagated to related artifacts based on the Dcoupling value. Since the artifacts are related one another directly (adjacently) or indirectly (through intermediate artifacts) the change propagation should be calculated for the above two cases.

Let A and B are two artifacts that are related adjacently and attributes of an artifact B access attributes of an artifact A. In this paper an attribute is considered as a feature of an artifact. The Dcoupling between artifacts A and B is defined as follows.

Dcoupling = number of links with respect to attributes of A from B/Total number of possible links from B to A. The denominator can be taken as the total number of attributes exists in A, since this many maximum links can be made. In the above equation, a link is defined as a call made by artifact B with respect to an attribute of artifact A. It is irrespective of number of calls made to each attribute. That is, all calls of an attribute constitute a link even though if it is called more than once by an attribute of class B. For example, consider a class A has three attributes (a1, a2 and a3) and B has five (b1, b2, b3, b4 and b5) attributes. The attribute a2 is called three times by attributes of class B and attribute a3 is called two times by attributes of class B. The Dcoupling between A and B for this case is 2/3. Higher value of Dcoupling indicates higher strength of dependency (coupling) between the artifacts. A design change is propagated to related artifacts based on the Dcoupling value. For example, a design change is propagated to related artifacts whose Dcoupling value is more than or equal to the threshold (say 0.5) value.

The above procedure is used to estimate Dcoupling value between adjacent artifacts. A more general approach would be to estimate the ramifications due to a single change. Therefore, the following method is used to compute combined Dcoupling value for the artifacts that are connected through intermediate artifacts in more than one path. The combined Dcoupling value is the probability that the end effect will arise, regardless of the path. This can be calculated using probability lemmas. While calculating the combined Dcoupling value it is to be noted that the events are not mutually exclusive. Therefore, the following formulas (1 and 2) are used to estimate the combined Dcoupling values [4].

$$\text{Dcoupling}_{p,q} \cap \text{Dcoupling}_{p,r} = \text{Dcoupling}_{p,q} \text{ X } \text{Dcoupling}_{p,r} \quad (1)$$

$$\text{Dcoupling}_{p,q} \cup \text{Dcoupling}_{p,r} = \text{Dcoupling}_{p,q} + \text{Dcoupling}_{p,r} - \left(\text{Dcoupling}_{p,q} \text{ X Dcoupling}_{p,r}\right) = 1 - \left[\left(1 - \text{Dcoupling}_{p,q}\right) \text{X} \left(1 - \text{Dcoupling}_{p,r}\right)\right] \quad (2)$$

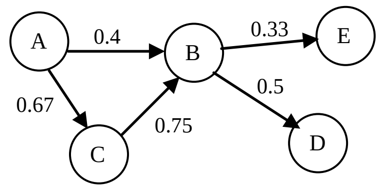

Figure 1. Example URA graph of artifacts

Dcoupling values for the adjacent artifacts and the combined Dcoupling values taking intermediate artifacts into account is computed. Example design diagram shown in figure 1 is an URA graph. The values on the top of links represent Dcoupling values between various artifacts that are adjacent. The combined Dcoupling value at artifact B in the figure 1 is calculated as follows:

$$\text{Dcoupling}_{a,b} = 1 - [(1 - \text{Dcoupling}_{a,b}) \text{ X } (1 - (\text{Dcoupling}_{a,c} \text{ X } \text{Dcoupling}_{c,b}))] = 0.70$$

Using these Dcoupling values, n by n dependency matrix is constructed. Where n is the number of artifacts in the design. This dependency matrix cell values represent design change propagation probabilities between different artifacts.

3. Dependency Oriented Complexity Metrics

The proposed dependency oriented complexity metrics are presented in this section. All the three metrics make use of the design change propagation probabilities of dependency matrix of a design. The Dcoupling values represent design change propagation probabilities between different artifacts. The Dcoupling values for a particular given design are computed as specified in section 2 and a dependency matrix of size n by n is constructed. This dependency matrix is used in computing the proposed metrics.

3.1 Dependency Oriented Complexity Metric for the Structure (DOCMS(R))

The complexity of structure (architecture) mainly depends on coupling between artifacts and the strength of coupling (Dcoupling). The design change is propagated to related artifacts based on the value of Dcoupling. High value of Dcoupling means high probability of change propagation which indicates high complexity of the structure. Considering this view, a metric is proposed for the structure. The dependency oriented complexity metric for the structure (architecture) DOCMS(R), which considers rippling effects is defined as follows.

$$\text{DOCMS(R)} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \text{DCPP}_{ij}}{(n^2 - n)} \quad (3)$$

The n in the equation is number of artifacts in the URA graph. The summation is done for all the values of i and j except for i equals j (diagonal elements are not summed up). Using the dependency matrix and making use of the equation 3 the DOCMS(R) is computed.

3.2 Dependency Oriented Complexity Metric for an Artifact Causing Ripples (DOCMA(CR))

The bad smell shotgun surgery occurs when every time a change is made to an artifact, change is propagated to many different artifacts, which leads to changes in those artifacts [12]. To detect which class/artifact is creating ripples and propagating change to different artifacts, a metric is proposed. The dependency oriented complexity metric for an artifact causing ripples is defined as follows.

$$\text{DOCMA}_i\text{(CR)} = \frac{\sum_{j=1}^{n} \text{DCPP}_{ij}}{(n-1)} \quad (4)$$

Where n is the number of artifacts in the URA graph and the summation is performed for $i \neq j$. DOCMA_i(CR) is calculated for different artifacts by varying i from 1 to n. This metric value above a threshold (say 0.5) for an artifact indicates that the artifact is causing more ripples and the design change in this artifact is propagated to different artifacts. This indicates the presence of design defect caused by shotgun surgery bad smell. The artifacts to which the design change is propagated can be known from the dependency matrix.

3.3 Dependency Oriented Complexity Metric for an Artifact Affected by Ripples (DOCMA(AR))

The divergent change bad smell is present in the design when changes for different reasons in different artifacts have propagated to an artifact, which leads to many changes in that artifact [12]. To detect the class/artifact which is affected by many ripples coming from different directions a metric is proposed. The dependency oriented complexity metric for an artifact affected by ripples is defined as follows.

$$\mathrm{DOCMA}_j(\mathrm{AR}) = \frac{\sum_{i=1}^{n} \mathrm{DCPP}_{ij}}{(n-1)} \qquad (5)$$

Where n is the number of artifacts in the URA graph and the summation in the equation is performed for $j \neq i$. DOCMA$_j$(AR) is calculated for different artifacts by varying j from 1 to n. This metric value above a threshold (0.5) for an artifact indicates that the artifact is affected by more ripples coming from different artifacts and this indicates the presence of design defect caused by the divergent change bad smell. Using dependency matrix the artifacts from which design changes are propagated to this artifact can be known.

4. Experimental Results

Two different design defects indicated by the presence of two bad smells shotgun surgery and divergent change are considered for detection. Four cases (hypothetical example designs) are considered for detecting design defects. The DOCMA(CR) metric value for an artifact above a threshold indicates that the artifact is causing more ripples and the design change in this artifact is propagated to different artifacts. This indicates the presence shotgun surgery bad smell in the design. The DOCMA(AR) metric value for an artifact above a threshold indicates that the artifact is affected by more ripples coming from different artifacts and this indicates the presence of divergent change bad smell. In this paper a threshold value of 0.5 is considered for DOCMA(CR) and DOCMA(AR).

Case 1: Design 1 with six artifacts and Dcoupling values between adjacent artifacts as shown in figure 2 is considered.

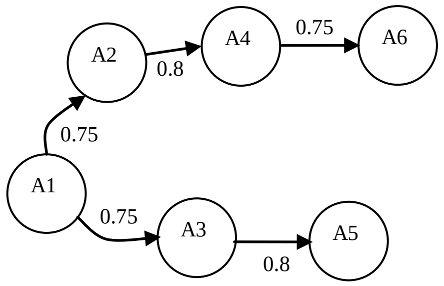
$$\begin{tikzcd}
 & A2 \arrow[r, "0.8"] & A4 \arrow[r, "0.75"] & A6 \\
A1 \arrow[ur, "0.75"] \arrow[dr, "0.75"'] & & & \\
 & A3 \arrow[r, "0.8"] & A5 &
\end{tikzcd}$$

Figure 2. URA Graph of design 1

The combined Dcoupling values for the artifacts that are connected through intermediate artifacts in more than one path of the design 1 are estimated using the equations given in section 2. A dependency matrix of size n by n is constructed using these values. These values indicate design change propagation probabilities (DCPP) between artifacts. The design change propagation probabilities for the design 1 (case 1) are given in table 1.

	A1	A2	A3	A4	A5	A6
A1	1	0.75	0.75	0.6	0.6	0.45
A2		1		0.8		0.6
A3			1		0.8	
A4				1		0.75
A5					1	
A6						1

Table 1. Dependency matrix for design 1
(Design change propagation probabilities)

The DOCMA(CR) and DOCMA(AR) values for different artifacts of the design 1 are computed using the equations 4 and 5 respectively. For computing these metrics the dependency matrix given in table 1 is used. These metrics values are given in table 2.

Metric / Artifact	DOCMA(CR)	DOCMA(AR)
A1	**0.63**	0
A2	0.28	0.15
A3	0.16	0.15
A4	0.15	0.28
A5	0	0.28
A6	0	0.36

Table 2. Computed values of DOCMA(CR) and DOCMA(AR) for design 1

The DOCMA(CR) value of artifact A1 is 0.63. This value is above threshold value of 0.5. This indicates the presence of design defect caused by the shotgun surgery bad smell. The bad smell shotgun surgery occurs when every time a change is made to an artifact, change is propagated to many different artifacts, which leads to changes in those artifacts. The values of DOCMA(AR) for different artifacts are below threshold (table 2). This indicates that the divergent change bad smell is not present in the design.

Case 2: Design 2 with six artifacts and Dcoupling values between adjacent artifacts as shown in figure 3 is considered.

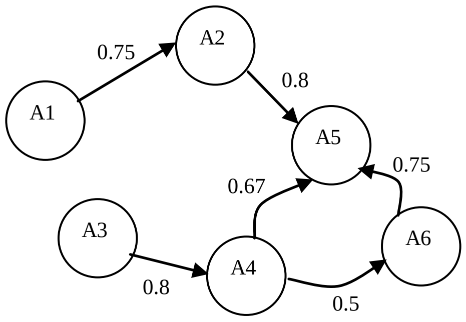

Figure 3. URA Graph of design 2

The combined Dcoupling values for the artifacts that are connected through intermediate artifacts in more than one path for the design 2 are estimated using the equations given in section 2. A dependency matrix of size n by n is constructed using these values. The dependency matrix for design 2 is given in table 3.

	A1	A2	A3	A4	A5	A6
A1	1	0.75			0.6	
A2		1			0.8	
A3			1	0.8	0.67	0.40
A4				1	0.79	0.5
A5					1	
A6					0.75	1

Table 3. Dependency matrix for design 2
(Design change propagation probabilities)

The DOCMA(CR) and DOCMA(AR) values for different artifacts of the design 2 are computed using the equations 4 and 5 respectively. For computing these metrics the dependency matrix given in table 3 is used. These metrics values are given in table 4.

Metric / Artifact	DOCMA(CR)	DOCMA(AR)
A1	0.27	0
A2	0.16	0.15
A3	0.374	0
A4	0.258	0.16
A5	0	**0.722**
A6	0.15	0.18

Table 4. Computed values of DOCMA(CR) and DOCMA(AR) for design 2

The values of DOCMA(CR) for different artifacts are below threshold. This indicates that the shotgun surgery bad smell is not present. Since the DOCMA(AR) value of artifact A5 is above threshold, indicates the presence of design defect caused by the divergent change bad smell. The divergent change bad smell is present in the design when changes for different reasons in different artifacts have propagated to an artifact, which leads to many changes in that artifact.

Case 3: Design 3 as shown in figure 4 is considered.

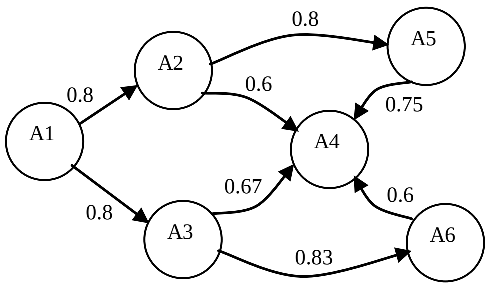

Figure 4. URA Graph of design 3

Making use of the computed Dcoupling values, dependency matrix for the design 3 is constructed. The dependency matrix for design 3 is given in table 5.

	A1	A2	A3	A4	A5	A6
A1	1	0.8	0.8	0.93	0.64	0.66
A2		1		0.84	0.8	
A3			1	0.83		0.83
A4				1		
A5				0.75	1	
A6				0.6		1

Table 5. Dependency matrix for design 3
(Design change propagation probabilities)

The DOCMA(CR) and DOCMA(AR) values for different artifacts of the design 3 are computed using the equations 4 and 5 respectively. For computing these metrics the dependency matrix given in table 5 is used. These metrics values are given in table 6.

Metric / Artifact	DOCMA(CR)	DOCMA(AR)
A1	**0.766**	0
A2	0.328	0.16
A3	0.332	0.16
A4	0	**0.79**
A5	0.15	0.29
A6	0.12	0.3

Table 6. Computed values of DOCMA(CR) and DOCMA(AR) for design 3

The DOCMA(CR) value of artifact A1 is 0.766. This value is above threshold value of 0.5. This indicates the presence of shotgun surgery bad smell in the design. The DOCMA(AR) value of artifact A4 is 0.79 which is above threshold indicating the presence of divergent change bad smell in the design. In this design two bad smells are detected.

Case 4: Design 4 as shown in figure 5 is considered.

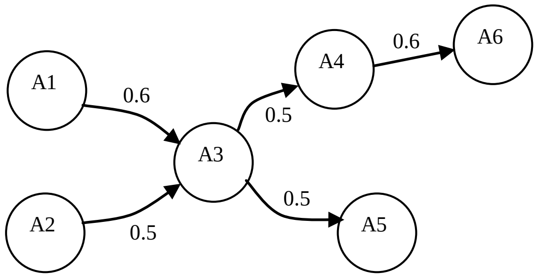

Figure 5. URA Graph of design 4

Dependency matrix for the design 4 is constructed, by making use of the computed Dcoupling values. The dependency matrix for design 4 is given in table 7.

	A1	A2	A3	A4	A5	A6
A1	1		0.6	0.3	0.3	0.18
A2		1	0.5	0.25	0.25	0.15
A3			1	0.5	0.5	0.3
A4				1		0.6
A5					1	
A6						1

Table 7. Dependency matrix for design 4
(Design change propagation probabilities)

The DOCMA(CR) and DOCMA(AR) values for different artifacts of the design 4 are computed using the equations 4 and 5 respectively. For computing these metrics the dependency matrix given in table 7 is used. These metrics values are given in table 8.

Metric / Artifact	DOCMA(CR)	DOCMA(AR)
A1	0.28	0
A2	0.23	0
A3	0.26	0.22
A4	0.12	0.21
A5	0	0.21
A6	0	0.25

Table 8. Computed values of DOCMA(CR) and DOCMA(AR) for design 4

Since both the metrics values for different artifacts are below threshold, the shotgun surgery and divergent change bad smells are not present in the design 4.

The experimental results for all the four cases have been given in table 9. For all the four cases DOCMS(R) metric is also computed. The DOCMA(CR) and DOCMA(AR) values given in table 9 are computed maximum values for any artifact.

Metric	**Case 1**	**Case 2**	**Case 3**	**Case 4**
DOCMS(R)	0.203	0.202	0.283	0.148
DOCMA(CR)	0.63	0.374	0.766	0.28
DOCMA(AR)	0.36	0.722	0.79	0.25
Edges/ Nodes	0.83	1	1.33	0.83

Table 9. Experimental results

The results presented in table 9 clearly indicate that the design which contains more defects has higher value of DOCMS(R). Case 3 (Design 3) has two defects, whereas case 4 has no defects. Case 1 and case 2 have one defect each. The complexity of case 3 is more than the other three cases as it contains two defects. This is indicated by the DOCMS(R) metric value (table 9). The case 4 value is less than the other three cases since it does not contain any defect. Generally the complexity of structure depends on the presence of design defects. The DOCMS(R) metric value successfully indicated with variation the complexity of structure depending on the number of defects present in the program structure (architecture). Knowing the complexity of the design helps to estimate required maintenance efforts and gives way for refactorings.

The high value of DOCMA(CR) above threshold (0.5) indicates the presence of design defect caused by shotgun surgery bad smell, whereas the high value of DOCMA(AR) above threshold (0.5) indicates the presence of design defect caused by the divergent change bad smell. The metric edges to nodes ratio is also computed for all the four cases. This metric computes the ratio of total number of direct interactions between the artifacts to total number of artifacts. This metric value is same for case 1 and case 4. Case 1 has a design defect, whereas case 4 does not have any defect. Hence, this metric value fails to differentiate between case 1 and case 4. Proposed metrics could differentiate properly the design with and without defects. The proposed metrics can be used to detect quantitatively the presence of design defects caused by shotgun surgery and divergent change bad smells. The detection is amicable for automation.

5. Related Work

In this section, related work on design defects detection strategies/methods/metrics, in particular, methods/metrics which cover design change propagation (ripple effect) between artifacts is given.

Paper [5] indicates that the CK metrics appear to be useful to predict class fault proneness during the design phase. Even though CK metrics [6] are useful in predicting class fault proneness, consider the metric CBO (Coupling between object classes), which is defined as a count of the number of other classes to which it (a class under consideration) is coupled. It is not indicating the amount (strength) of coupling between any two classes. Considering the number of discrete messages exchanged between classes, the god classes are identified using link analysis method [7]. The proposed method in this paper [7] is used to detect design defect indicated by the presence of god class bad smell. Our proposed metrics are aimed to detect design defects indicated by the presence of shotgun surgery and divergent change bad smells.

The paper [8] describes the ripple effect metric. It considers its applicability as a software complexity measure for object oriented software. It is mentioned that this approach has potential to improve the stability and efficiency of object oriented software and cut the cost of software maintenance. But the metrics proposed in this paper are aimed at detecting design defects caused by shot gun surgery and divergent change bad smells, in addition to complexity metric for structure. List of metric based detection strategies for capturing flaws of object oriented design are defined in paper [3]. Paper [9] investigated the construction of probabilistic decision models based on coupling measurement to support impact analysis. It provides an ordering of classes where ripple effects are more likely. A metric for measuring the class weakness for object oriented software is proposed in paper [10]. Inter-class weakness is affected by the interconnection of the class over other classes, and increases if the dependency of the class is more. The ripple effect also contributes to the dependency and this effect has also been considered in this paper. Even though, the papers ([9], [10]) have considered ripple effect, but, they have not correlated the ripple effect results with design defects/bad smells. The proposed metrics are intended to detect design defects caused by the presence of shotgun surgery and divergent change bad smells. Quantitative detection of design defects using metrics enables to perform appropriate refactorings to improve the quality of design.

6. Conclusions and Future Directions

Three metrics are proposed in this paper. They are, dependency oriented complexity metric for structure (DOCMS(R)), dependency oriented complexity metric for an artifact causing ripples (DOCMA(CR)), and dependency oriented complexity metric for an artifact affected by ripples (DOCMA(AR)). DOCMS(R) metric value indicated the higher complexity for the structure due to presence of design defects. In case 1 high DOCMA(CR) metric value indicated the presence of shotgun surgery bad smell, whereas in case 2 the high DOCMA(AR) metric value indicated the presence of divergent change bad smell. In Case 3 two bad smells are present and detected using the proposed metrics. Detecting the bad smells (design defects) allows the designer to apply appropriate refactorings to make software maintainable and to improve the quality of software.

Appropriate refactorings meant for shotgun surgery and divergent change bad smells need to be applied to see the changes in metric values after refactorings. The metrics have to be further validated applying on industrial case studies.

References

[1] Ladan Tahvildari, Kostas Kontogiannis, “A metric-Based Approach to Enhance Design Quality Through Meta-Pattern Transformations”, Proc. of the Seventh IEEE European Conf. on Software Maintenance and Reeng. (CSMR’03), pp. 183-192, 2003.

[2] Tom Mens, Tom Tourwe, “A Survey of Software Refactoring”, IEEE Trans. Software Eng., vol.30, no. 2, pp.126-139, Feb. 2004.

[3] R. Marinescu, “Detection Strategies: Metrics-Based Rules for Detecting Design Flaws”, In Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM), pp.350-359, September 2004.

[4] A Ananda Rao, D. Janaki Ram, “Software Design Versioning using Propagation Probability Matrix”, In Proceedings of Third Int’l Conf. on Computer Applications (ICCA 2005), Yangon, Myanmar, March 2005.

[5] V.R.Basili, L.C.Briand, and W.I.Melo, “A Validation Of Object Oriented Design Metrics as Quality Indicators”, IEEE Trans. Software Eng., vol. 22, no.10, pp.751-761, October 1996.

[6] Shyam R. Chidamber, Chris F. Kemerer, “A Metrics Suite for Object Oriented Design”, IEEE Trans. Software Eng., vol.20, no.6, pp. 476-493, June 1994.

[7] Alexander Chatzigeorgiou, Spiros Xanthos, George Stephanides, “Evaluating Object-Oriented Designs with Link Analysis”, Proceedings of the 26th IEEE Int’l Conf. on Software Engineering (ICSE ’04), pp. 656-665, 2004.

[8] Haider Bilal, Sue Black, “Computing Ripple Effect for Object Oriented Software”, Quantitative Approaches in Object Oriented Software Engineering (QAOOSE) Workshop, Nantes, France, July 2006.

[9] Lionel C. Brand, Jurgen Wust, Hakim Lounis, “Using Coupling Measurement for Impact Analysis in Object-Oriented Systems”, Proceedings of the IEEE Int’l Conf. on Software Maintenance (ICSM‘99), pp. 475-482, 1999.

[10] Jitender Kumar Chhabra, K.K.Aggarwal, “Measurement of Intra-Class & Inter-Class Weakness for Object-Oriented Soft-

ware", Proceedings of the Third IEEE International Conference on Information Technology: New Generations (ITNG'06), pp. 155-160, 2006.

[11] S. Srinath, URA: "A Paradigm for Context Sensitive Reuse", A Thesis of Master of Science & Engineering, Indian Institute of Technology, Madras, India, April 1998.

[12] Martin Fowler, K.Beck, J.Brant, W.Opdyke, D.Roberts, "Refactoring: Improving the Design of Existing Code". Addison- Wesley, NY, 1999.