International Journal of Information Technology and Knowledge Management
July-December 2011, Volume 4, No. 2, pp. 501-509

COUPLING COMPLEXITY NORMALIZATION METRIC-AN OBJECT ORIENTED PERSPECTIVE

Amitabha Yadav[1] & Raees Ahmad Khan[2]

Over the last decades, software quality attributes have been widely studied. In contrast, less attention has been paid to the field of software reliability. Complexity is a major factor of software reliability which degrades the performance of reliability. High complexity makes the system unreliable. The paper proposes a new method to improve reliability of object oriented design by normalizing complexity which is closely correlated with coupling. Coupling increases complexity and thus decreases reliability. A coupling complexity normalization (CCN) metric is proposed to minimize complexity of object oriented software design. An algorithm has been developed to normalize complexity. The proposed metric (CCN) is studied with the help of OSCAR case study. Coupling Complexity Relationship among classes has been calculated. Results obtained on implementing CCN is further compared with coupling complexity measure using existing standard metric WCC. It is seen that the normalized complexity using CCN has given much lesser value than the WCC.

Keywords: Coupling Complexity Normalization, Coupling Complexity Relationship, Reliability, Coupling, Weighted Class Complexity, Eigen Values, Eigenvectors.

1. Introduction

Towards moving 21st century, software becomes a driver for everything from elementary education to genetic engineering. Software developers analyze reliability, maintainability and complexity of software through software measurement. Software is used to make decisions but on the other hand it is to believe that software can be unreliable as human beings. Dependency and requirements on computer increases the difficulties and complexities. Due to increase in dependency, the size and complexity of system has grown. According to literature survey high coupling decreases reliability of software and makes the software unreliable [1]. A little work has been reported to establish a correlation between OO software design constructs and quality attributes such as reliability, maintainability, dependability and so on [2]. Highly coupled modules increases complexity of software by making software unreliable. So it is necessary to consider reliability of software at early stage of development. Reliability is a major concern for every field such as in oil and gas industry, Software indusry, Health industry, Aircraft indusrty and so on [3].

There are several approaches to estimate complexity of software but none of them have been accepted as a true measure of complexity of a class diagram [4]. Object oriented perspective is one of the most significant ways to quantify reliability of software by controlling object oriented constructs. Object oriented design provides a novel approach for problem solving using models around real world entities. Most of the software projects are shifting towards object oriented design because of only that design phase of a software development life cycle is the only phase in which structure of a software is made available. Design phase is the vehicle to produce high quality object oriented application that brings blue print into the reality [5]. Object oriented language supports coupling thereby sharing the data members and member function among the class hierarchy. Coupling is the degree to measure the software reliability.

Coupling is one of the major principal of object oriented software which contributes lot to software industry and is a characteristic of software design among internal attributes such as modularity, low coupling, high cohesion, encapsulation, inheritance etc. From last two decades lots of metrics has been proposed ranges from cohesion to coupling in object oriented software. However, coupling has a negative impact on software reliability [5]. High coupling results in bad software design which increases complexity of software. Software metrics are used to measure the complexity of program. The greater the number of coupled classes in class design, greater will be the complexity of class hierarchy. Coupling in a class is supposed to be more sensitive and fault-prone because of shearing among classes at deeper level. It increases common cause failure rates.Common cause failures rates are a specific class of dependent failures that affect complex system [6]. Thereby, increasing the demand for reliable software. To minimize complexity of software it is necessary to control coupling of object oriented software. Coupling is closely

[1]DIT, Babasaheb Bhimrao Ambedkar University, Lucknow, India
[2]DIT, Babasaheb Bhimrao Ambedkar University, Lucknow, India
E-mail: [1]amitabha.engg@yahoo.com, [2]khanraees@yahoo.com

related with reliability. To improve reliability of software, coupling should be taken into consideration to minimize complexity of software [7]. Complexity is a main factor of reliability. To get high reliable software it is necessary to decrease or maintain complexity of software by controlling coupling.

The paper introduces terminology in section 2. Section 3 presents algorithm. Case study for WCC metric and coupling complexity normalization are discussed in section 4. Comparative study between two metrics has been done in Section 5. Section 6 includes finding. At last paper concludes in section 7.

2. TERMINOLOGIES INTRODUCED

Software reliability break, coupling complexity relationship and coupling complexity normalization are three terminologies introduced.

2.1 Definition 1: Software Reliability Break (SRB)

Software reliability break is that point where system ceases to do any function. Software reliability break is defined as the difference between time taken by a system to complete a function and time taken by a system to stop a function before completion. Reliability breaks by leaking or modifying information illegally which makes the system completely unreliable. Change coupling affect software reliability to some extent because change coupling correlates with software defects [8], which leads to software reliability break.

It is defined as

$$SRB = \left\{\begin{array}{c}\text{Total time to}\\ \text{perform a function}\end{array}\right\} - \left\{\begin{array}{c}\text{Time where system}\\ \text{stops to perform a function}\end{array}\right\}$$

$$SRB = \{TT\} - \{TS\}$$

Where,

SRB = Software Reliability Break

TT = Total time taken by a function to complete

TS = Time taken by function to stop to perform a function in between

2.2 Definition 2: Coupling Complexity Relationship (CCR)

Coupling complexity relationship is defined as a relationship among classes which increases complexity of a system as the amount of coupling increases in software system. Coupling and complexity are closely correlated with each other. Internal attributes influence directly or indirectly software reliability [9]. One such attribute appears to be coupling. Coupling is the shearing among software modules. As shearing word poses enormous faults to reliability break. Coupled class is more error prone [10] [11]. Due to sharing nature, information may leak either directly or indirectly from user domain to any unauthorized party. Although coupling seems to influence software reliability by influencing complexity.

2.3 Definition 3: Coupling Complexity Normalization (CCN)

Coupling complexity normalization is defined as the metric to minimize complexity by controling coupling of software system. Coupling complexity normalization is a proposed metric in this paper. It is a new key term which normalizes complexity. Normalization is a minimization technique. CCN gives one way to minimize complexity of objects oriented designs. A case study has been taken to evaluate the complexity in design phase. PageRank algorithm is used to measure the relationship among classes [12]. PageRank algorithm is an algorithm in web mining. It is used to measure the relationship between classes. The algorithm is proposed in 1998 by Sergey Brin and Lawerence Page [12]. Similarly here, classes are taken as web pages and links among classes as coupling relationship among classes.

It is defined as

$$CCN = \sum_{i=1}^{n} Ci$$

3. ALGORITHM

Coupling complexity normalization metric is used to measure complexity of OO design by normalizing procedure. An algorithm is proposed to measure overall complexity of design is as follows:

Step 1: To analyze dependence of class diagram and get correspondence class dependence diagram (CDD).

Step 2: Transform the class dependence diagram (CDD) to its matrix. Matrix R is used if there is a relationship from class i to j

then

$$R(i, j) = 1$$

Otherwise $R(i, j) = 0$

Step 3: Calculate stochastic transition matrix 'S' by using given formulae

$$S(i, j) = R(i, j) / \sum_{k=1}^{15} R(k, i)$$

Step 4 : Compute eigenvalue 'E' and eigenvector 'V' of S(i, j).

Step 5: Save the eigenvector to 'EV' corresponding to the maximum eigenvalue.

Step 6: Normalize 'EV' to get complexity normalization matrix vv..

Step 7: Coupling Complexity Relationship (CCR) per class is calculated.

Step 8: Sum of all Coupling Complexity Relationship (CCR) will give overall CCN.

4. CASE STUDY

OSCAR is Object Oriented Scene AnimatoR produces high quality film and video sequences of the result of scientific and engineering calculation and experiments [13]. OSCAR automates the creation, control, and management of 3-D computer generated animation Sequences. Using an object oriented script language as its user interface; OSCAR controls analysis, modeling, rendering, display and filming. Interfaces have been developed for scientific analysis programs in the areas of molecular modeling, robotic, mechanism analysis and fluid mechanics. The object oriented design has produced a system that lends to interfacing with existing and future applications. The OSCAR application demonstrates how an object oriented design can be successfully implemented with a non objected language. OSCAR is written in C using an object oriented development environment that includes macros and run time support for inheritance, instancing, and message passing. OSCAR includes an interpreter that permits users to sending messages.

OSCAR case study has been used to for calculating complexity of object oriented design by two different approaches. One approach is already developed metric and the other one is proposed metric [14]. Weighted class complexity (WCC) metric is already developed metric and is used to measure complexity of overall class hierarchy. The other metric is proposed one i.e coupling complexity normalization (class complexity normalization). Quantitative estimation of both metrics has been calculated and than compared with each other.

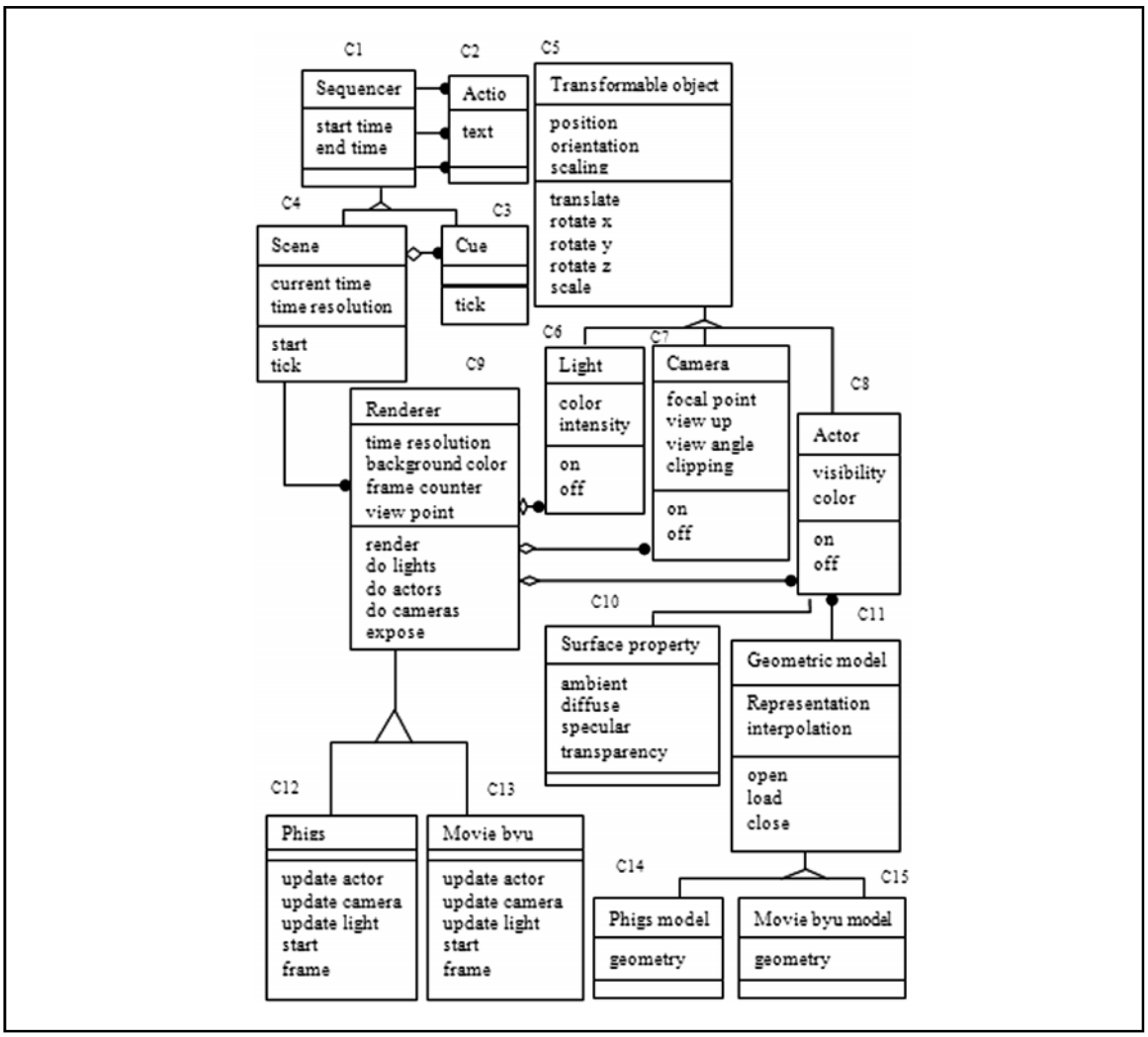

Fig.1: Class Diagram for OSCAR System

4.1 Complexity Estimation Using WCC

Coupling and complexity are closely related with each other. Both have negative impact on reliability. The more number of coupling, the lower is the reliability of software and greater the complexity [15]. Increase in coupling should cause an increase in WCC [16]. It is known that higher the cohesion, the better the design and therefore higher cohesion should cause decreases in WCC. Greater inheritance increases design complexity and makes the system difficult in implementation and maintenance [17]. To make the system more reliable it is necessary to control coupling and complexity. Weighted class complexity metric is used to measure the overall complexity of a class diagram OSCAR. Weighted class complexity is defined as below.

$$WCC = (RFC * Level) + LCOM$$
$$RFC = WMC + CBO$$

Where, WCC = Weighted class complexity
RFC = Response for a class
Level = Depth of Inheritance
WMC = Weighted method per class
CBO = Coupling between Object

Table 1
Weighted Class Complexity/Class

Class	WMC	CBO	RFC	LEVEL	LCOM	WCC
C1	0	5	5	0	0	0
C2	0	3	3	0	0	0
C3	1	3	4	0	1	1
C4	2	3	5	1	2	7
C5	5	2	7	1	5	12
C6	2	2	4	1	1	5
C7	2	2	4	1	1	5
C8	2	4	6	1	1	7
C9	5	4	9	0	5	5
C10	0	1	1	0	0	0
C11	3	3	6	0	3	3
C12	5	1	6	1	1	7
C13	5	1	6	1	1	7
C14	0	1	1	1	0	1
C15	0	1	1	1	0	1
						$\sum_{i=1}^{n}(WCC)i$ 61

Weighted class complexity is calculated by taking an average of sum of weighted class complexity per class.

$$WCC = \sum_{i=1}^{n}(WCC)\, i / n$$

Where, n = number of classes
WCC = 61/15 = 4.06
WCC = 4.06

4.2 Complexity Estimation Using CCN

A class is coupled to another class if methods of one class use methods or attribues of the other class. According to algorithm, relationship between different classes has been calculated in matrix form shown in Table 1. The relationship is being developed on the bases of OSCAR case study presented in Fig 1. Firstly, analyze the dependence of class diagram to get CDD (Class Dependence Diagram). Class design is firstly transformed into the class dependence diagram (CDD) to its matrix.

Matrix R is developed if there is a relationship from class i to j.

Then $R(i, j) = 1$
Otherwise $R(i, j) = 0$

Table 2
Matrix R Corresponding to CCD of OSCAR

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
C2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
C4	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0
C5	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
C6	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
C7	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
C8	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0
C9	0	0	0	1	0	1	1	1	0	0	0	1	1	0	0
C10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C11	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
C12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
C13	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
C14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C15	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

After calculating class dependence diagram, stochastic transition matrix is calculated. CDD of OSCAR case study is transformed into matrix 'S' according to step 2 of algorithm. Table 2 presents stochastic transition matrix and is being calculate on the basis of given formulae.

$$S(i, j) = R(i, j) / \sum_{k=1}^{15} R(k, i)$$

Table 3
Stochastic Transition Matrix 'S'

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	0	1	0.5	0.33	0	0	0	0	0	0	0	0	0	0	0
C2	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C3	0.33	0	0	0.33	0	0	0	0	0	0	0	0	0	0	0
C4	0.33	0	0.5	0	0	0	0	0	0.16	0	0	0	0	0	0
C5	0	0	0	0	0	0.5	0.5	0.25	0	0	0	0	0	0	0
C6	0	0	0	0	0.33	0	0	0	0.16	0	0	0	0	0	0
C7	0	0	0	0	0.33	0	0	0	0.16	0	0	0	0	0	0
C8	0	0	0	0	0.33	0	0	0	0.16	1	0.33	0	0	0	0
C9	0	0	0	0.33	0	0.5	0.5	0.25	0	0	0	1	1	0	0
C10	0	0	0	0	0	0	0	0.25	0	0	0	0	0	0	0
C11	0	0	0	0	0	0	0	0.25	0	0	0	0	0	1	1
C12	0	0	0	0	0	0	0	0	0.16	0	0	0	0	0	0
C13	0	0	0	0	0	0	0	0	0.16	0	0	0	0	0	0
C14	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0
C15	0	0	0	0	0	0	0	0	0	0	0.33	0	0	0	0

Then, eigenvalue 'E' and eigenvector 'V' of Table 2 are computed. Then save the eigenvector to 'VV' corresponding to the max eigenvalue shown in Table 3. Lastly, VV is normalized with the help of Table 3. Calculate transpose of VV matrix which is a normalized matrix.

Table 4
Matrix 'E' of Eigenvalues

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C2	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C3	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C4	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C5	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C6	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C7	0.0	0.0	0.0	0.0	0.0	0.0	-0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.7	0.0	0.0	0.0	0.0	0.0	0.0
C10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0	0.0
C11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0
C12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	0.0	0.0	0.0
C13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0
C14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0
C15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Table 5: Matrix 'V' of Eigenvectors.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	-0.2	-0.5	0.2	-0.1	.07	-0.3	-.04	.09	0.8	-.07	.04	.00	-4.1	9.4	-1.6
C2	-0.9	-0.2	-01	-.08	-.06	-.50	.01	-.03	-.38	.07	-.03	-.00	-1.0	-2.4	-3.2
C3	-0.1	-0.3	0.1	-.01	-.01	0.4	-.02	.02	-0.3	-0.4	0.3	-0.2	1.9	-3.3	4.4
C4	-0.2	-0.3	.09	0.1	-.09	0.5	0.1	-0.1	-0.1	0.5	-0.4	0.4	2.2	3.8	-3.1
C5	-0.2	0.1	-0.3	-0.3	0.5	.08	-0.3	0.2	-.08	-0.3	-0.2	0.4	2.3	5.7	1.6
C6	-0.1	.05	-0.2	-.04	0.3	.01	0.2	-0.1	.05	0.1	0.2	-0.1	0.7	-.09	1.6
C7	-0.1	.05	-0.2	-.04	0.3	.01	0.2	-0.1	.05	0.1	0.2	-0.1	-0.7	.09	-1.6
C8	-0.3	0.3	-.03	-.51	-.50	-0.0	0.4	.06	.04	-0.1	-0.5	-0.4	-5.6	3.0	-7.0
C9	-0.5	.01	-0.5	0.6	-0.3	-0.1	-0.5	0.5	-0.0	0.4	-.04	-0.4	-8.6	-3.0	1.0

Table 5 Contd...

Table 5 Contd...

C10	-.09	.08	-.01	-.27	-.33	-0.0	-0.1	-.02	-.01	0.1	0.3	0.2	3.8	4.1	-2.1
C11	-..2	0.4	0.5	0.1	.08	2.7	-0.3	-0.6	0.0	-0.0	-0.1	-0.1	-2.8	4.3	2.6
C12	-.09	0.0	-0.1	0.2	-0.1	-0.1	.09	-0.1	.01	-0.2	.01	0.1	-1.9	-0.7	1.2
C13	-.09	0.0	-0.1	0.2	-0.1	-0.1	.09	-0.1	.01	-0.2	.01	0.1	1.9	0.7	-1.2
C14	-.09	0.1	0.2	.09	.07	4.6	0.1	0.2	-.02	.02	.09	.08	2.6	-2.7	-0.7
C15	-.09	0.1	0.2	.09	.07	4.6	0.1	0.2	-.02	.02	.09	.08	2.6	2.7	0.7

Table 6
Matrix 'V V' of Eigenvectors

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	1	0.5	0.27	.1	.07	-.30	-.04	.09	.83	-.07	.04	.00	-4.1	9.4	-1.6
C2	-.09	.91	-.11	-.08	-.06	-.50	.01	-.03	-.38	.07	-.03	-.00	-1.0	-2.4	-3.2
C3	-.19	-.34	.80	-.01	-.01	.48	-.02	.02	-.31	-.43	.30	-.28	1.9	-3.3	4.4
C4	-.29	-.38	.09	.47	-.09	.59	.12	-.16	-.15	.51	-.40	.43	2.2	3.8	-3.1
C5	-.29	.15	-.30	-.36	.51	.08	-.33	.21	-.08	-.38	-.28	.41	2.3	5.7	1.6
C6	-.19	.05	-.23	-.04	.31	.20	.21	-.19	.05	.16	.26	-.10	.7	-.09	1.6
C7	-.19	.05	-.23	-.04	.31	.01	.21	-.19	.05	.16	.26	-.10	-.70	.09	-1.6
C8	-.38	.31	-.03	-.51	-.50	-.00	.46	.06	.04	-.14	-.59	-.41	-5.6	3.0	-7.0
C9	-.58	.01	-.53	.60	-.30	-.14	-.58	.55	-.04	.44	-.04	-.49	-8.6	-3.0	1.0
C10	-.09	.08	-.01	-.27	-.33	-.00	-.11	-.02	-.01	.11	.37	.20	3.8	4.1	-2.1
C11	-.29	.44	.52	.13	.08	2.7	-.38	-.61	.05	-.02	-.11	-.13	-2.8	4.3	2.6
C12	-.09	.00	-.11	.21	-.13	-.12	.09	-.11	.01	-.22	.01	.15	-1.9	-0.7	1.2
C13	-.09	.00	-.11	.21	-.13	-.12	.09	-.11	.01	-.22	.01	.15	3.2	0.7	-1.2
C14	-.09	.16	.21	.09	.07	4.6	.13	.24	-.02	.02	.09	.08	2.6	1.4	-0.7
C15	-.09	.16	.21	.09	.07	4.6	.13	.24	-.02	.02	.09	.08	2.6	2.7	1.0

Table 5 depicts the relationship among classes. According to PageRank algorithm a page has high rank when the sum of the ranks of its back relationship is high. PageRank algorithm is defined that the rank of a page is sum of ranks of pages that points to it [12]. In the same way, it is imagine that classes as web pages and relationship among classes as a link among web pages weights has been calculated.

Table 7
Coupling Complexity Relationship/Class

C1	0.5065
C2	-0.0696
C3	0.7150
C4	-0.2283
C5	0.4054
C6	0.1685
C7	-0.0513
C8	0.3271
C9	-0.0184
C10	0.1032
C11	0.1649
C12	0.1854
C13	0.2590
C14	-0.6330
C15	-0.3244

Coupling Complexity Normalization calculates overall complexity of a class hierarchy. By using formulae coupling complexity Normalization (CCN) calculated as

$$CCC = \sum_{i=1}^{n} Ci$$

Coupling Complexity Normalization = 1.5100

Coupling Complexity Relationship is examined by plotting a graph between class complexity and class coupling. The Figure 2 indicates that complexity decreases while coupling increases. In some classes complexity is high but after second classes complexity starts decreasing and in some cases it is negative. Distribution of high and low complexity among nine classes is shown in Figure 3. High percent complexity class shows that rank of a class is high and the class is more error prone. 25% is highly complex class which indicates that number of relationship among classes is high that's why rank is high and makes the class highly sensitive which leads to software reliability break. There is less chances of software reliability break, in case of 4% distribution. Figure 2 and Figure 3 are constructed on the basis of coupling complexity relationship.

5. COMPARATIVE STUDY

To prove the claim that CCN may be used to compare with already developed metrics for the same designs of object

oriented software as well as designs of different object oriented software. WCC metric is already developed metric and is numerically estimated the complexity of an object oriented software design. CCN is a proposed metric and it also estimates numerical value of complexity of object oriented software design. These two metrics measures complexity for the same object oriented case study OSCAR. From the available detailed design documents, it is found that OSCAR case study has 15 classes, which are listed as:

Design (OSCAR) = (Sequencer, Action, Transformable object, Scene, Cue, Renderer, Light, Actor, Camera, Surface property, Geometric model, Phigs, Movie byu, Phigs Model, Movie byu model).

For implementation of the proposed approach any type of documents including Collaboration Diagrams, Sequence Diagrams, State Diagrams and Class Hierarchy are not required. The only thing needed is the detailed design of classes showing all the methods and attributes. WCC and CCN are compared with each other. Detailed comparative study is as discussed below:

WCC computation for OSCAR case study: On applying the procedure for calculating WCC for OSCAR design, the following result is obtained:

(a) Weighted class complexity calculates complexity of each class. This is done by using WCC metric.

(b) WCC metric is (RFC*Level) + LCOM

(c) RFC = WMC + CBO

(d) WCC = $\sum_{i=1}^{n}$(WCC)i / n = 61/15 = 4.06

Table 1 shows the value of metrics for each class. Various metrics such as RFC, CBO, WMC, Level and LCOM are calculated for each class present in the design hierarchy.

CCN computation for OSCAR case study: On applying the algorithm for calculating CCN for OSCAR design, the following results are obtained:

(a) Coupling class dependence matrix is obtained. If R(i, j) = 1 than C_i class is coupled with C_j class. If R (i, j) = 0 than C_i class is not coupled with C_j class.

(b) Stochastic transition matrix is developed to compute eigen values and eigenvectors. VV matrix is calculated on the basis of maximum eigen value to eigenvector. VV matrix and eigenvalues are multiplied to get a 15*1 matrix. This 15*1 matrix is coupling complexity relationship (CCR) matrix.

(c) Normalization procedure is than implemented on CCR matrix. Normalized matrix is calculated by dividing each element by taking square root of sum of square of each element.

(d) CCN is calculated by summation of weights of each classes.

$$CCN = C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7 + C_8 + C_9 + C_{10} + C_{11} + C_{12} + C_{14} + C_{15}$$
$$= 1.5100$$

Table 7 shows the weights of each class. Complexity is calculated by adding all the weights of the class.

Table 8
A Comparative Analysis of Metrics WCC and CNN

Analysis / Metrics	Classes	Complexity	Calculation Remarks
WCC	15	4.06	Complexity of WCC metric is greater than CCN metric
CCN	15	1.51	

Relative Results: A comparative analysis for both metrics of OSCAR design is presented in Table 8. Quantitative figures obtained from both metrics show that complexity computed by WCC metric is more than the complexity computed by CCN metric.

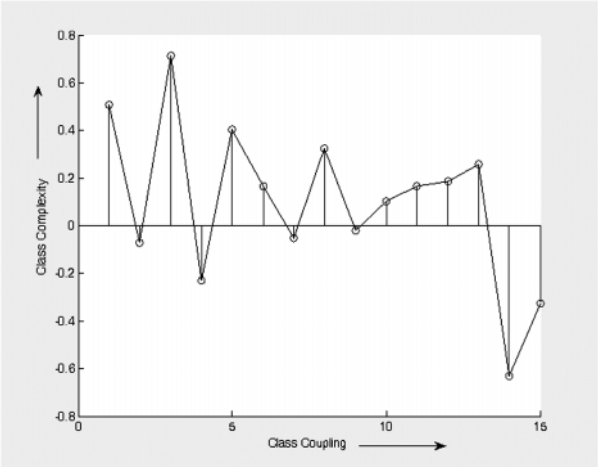

Fig.2: Coupling Complexity Relationship/Class

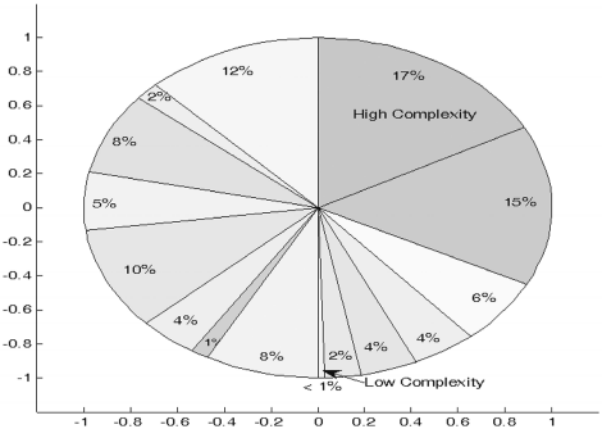

Fig.3: Distribution of Complexity Among all the Classes of a Class Design.

6. FINDINGS

The paper presents the estimation of software reliability by considering its factor complexity. Complexity is a factor of reliability and complexity of object oriented software can be maintained by controlling coupling. Coupling is closely correlated with reliability factor such as complexity. Therefore, complexity has been chosen as a major factor to increase reliability by normalization method.

The contribution of the work is summarized as follows:

- Three terminologies software reliability break (SRB), coupling complexity relationship (CCR) and coupling complexity normalization (CCN) has been introduced.
- CCN is taken as new metric to calculate complexity of a system.
- An algorithm is used to measure of object oriented software to normalize complexity.
- Object oriented SCene AnimatoR (OSCAR) case study has been used to examine the role of Object oriented construct coupling over complexity.
- Complexity of same class hierarchy is calculated for two metrics WCC and CCN. Calculation indicates that CCN complexity is lower than calculation of WCC metric.
- Coupling Complexity Relationship is calculated.
- Coupling Complexity Normalization metric is used to normalize the complexity among shared classes and is being calculated numerically.
- Numerical values of CCN justify their betterment over WCC metric.
- Complexity of object oriented software can be maintained by controlling design construct coupling to increase reliability of software.

7. CONCLUSION

Complexity is a major factor of software reliability. High complexity lead to high software reliability breaks by making the system unreliable. Object oriented design has been chosen for the analysis of software reliability factor complexity. High Coupling increases complexity and coupling is a major construct of object oriented design. High coupling and greater complexity both plays a role in software reliability break. To control software reliability breaks, it is believed to control coupling which normalizes complexity to some extent.

During development design phase develops basis of software. A weak base never inclines to produce reliable software. Complexity is one such attribute of reliability for which none of the complexity estimation approach have been accepted as a true measure of complexity of class design of object oriented software [4]. A metric CCN has been proposed and the algorithm for complexity calculation has been designed. The proposed metric is compared with already developed metric WCC. These two metrics are used to estimate numerical value of complexity for the same case study OSCAR.

REFERENCES

[1] R.Tripathi, and R.Mall, "Early Stage Software Reliability and Design Assessment", IEEE Computer Society, Proceedings of the 12th Asia Pacific Software Engineering Conference (APSEC'05), ISSN: 1530-1362, 2005.

[2] V. R. Hudli, L.C. Hoskins, and V. A. Hudli, "Software Metrics for Object Oriented Design", IEEE, ISBN: 0-8186-6565-3 pp:492-495,1994.

[3] X.Gao, J. Barabady and T. Markeset, "An Approach for Prediction of Prtroleum Production Facility Performance Considering Arctic Infulence Factors", Reliability Engineering and System Safety 95, Vol 95, No 8, pp.837-846, 2010.

[4] J. Xu, B.Randell, C.M.F. Rubira, and J. R.Stroud, "Towards an Object Oriented Approach to Software Fault Tolerance", IEEE, ISBN:!0-792-38069-X, pp. 226-232, 1995.

[5] A.Yadav and R. A. Khan, "Measuring Design Complexity–An Inherited Method Perspective", SIGSOFT Software Engineering Notes, 24 No.4,pp: 1-5, july 2009.

[6] A. Zitrou,T.Bedford and L. Walls, "Bayes Geometric Scaling Model for Common Cause Failure Rates", Reliability Engineering and System Safety, 95, No 2,pp.70-76, 2010.

[7] A. Yadav, and R. A.Khan, "Complexity:A Reliability Factor", IEEE International Advance Computing Conference (IACC-2009), Patiala, India, pp.2375-2378,6-7 March 2009.

[8] D. M. Ambros, M.Lanza, R.Robbes, "On the Relationship Between Change Coupling and Software Defects", 16th Working Conference on Reverse Engineering, pp.135-144, 2009.

[9] Y. M. Liu and I.Traore, "Empirical Relation Between Coupling and Attackability in Software Systems: A Case Study on DOS", Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security, ACM, ISBN:1-59593-374-3,PP.1-2,2006.

[10] T. Kamiya and S.Kusumoto, "Prediction of Fault-Proneness at Early Phase in Object Oriented Development", Object-Oriented Real-Time Distributed Computing, (ISORC '99) ISBN: 0-7695-0207-5, pp. 253 – 258, 1999.

[11] L. YU. Stephen, R. Sahach, K. Chen, and S. Ramaswamy, "Coupling Measurement in Multi –Kernel-Based Software with Its Application to Darwin", International Journal of Intelligent Control and Systems, 13, No 2,pp.109-119,June 2008.

[12] F. Li., T.Yi, "Apply Page Rank Algorithm to Measuring

Relationship's Complexity", IEEE, DOI 10.1109/ PACIIA.2008.309, ISBN: 9780769534909,pp. 914-917, 2008.

[13] YU.Liguo, R. Stephen, K. C. Svhach, and S. Ramaswamy, "Coupling Measurement in Multi Kernel Based Software with Its Application to Darwin", International Journal of Intelligent Control and Systems,13,No 2,pp.109-119,June 2008.

[14] R.A. Khan, K. Mustafa, and S.I. Ahson, "An Empirical Validation of Object Oriented Design Quality Metrics", King Saud Univ.19, Comp & Info Sci. Riyad, pp. 1-16,Dec 2007.

[15] A. Agrawal, and R.A. Khan, "A Vulnerability Metric for the Design Phase of Object Oriented Software", In Proceedings of IC3 (1)'2010, Elsevier, pp.328~339, 2010.

[16] K.F. Gerould, C.Theory, and A.Einstein, "Measuring Complexity", John Wiley and sons Inc., pp.54-78,2006,.

[17] M. Andersson, and P. Vestergren, "Object Oriented Design Quality Metrics", Uppsala Master,s Thesis in Computer Science, 276, ISSN 1100-1836, pp.1-67, June 2004.