An Empirical Investigation of the Correlation between Package-Level Cohesion and Maintenance Effort

Waleed Albattah
Department of Information Technology
Qassim University
Qassim, Saudi Arabia

***Abstract*—The quality of the software design has a considerable impact on software maintainability. Improving software quality can reduce costs and efforts of software maintenance. Cohesion, as one of software quality characteristics, can be used as an early indicator for predicting software maintenance efforts. This paper improves Martin's cohesion metric, which is one of the well-known and well-accepted cohesion metrics. The strong correlation found between package cohesion, using our proposed metric, and maintenance efforts shows the improvement made on measuring cohesion, and how it would be for predicting maintenance efforts. The experimental study included data from four open source Java software systems. The results show that the package cohesion is good and low maintenance is required.**

***Keywords*—package; cohesion; metric; maintenance effort; maintainability; software; measurements**

I. Introduction

Software maintainability refers to the ease of maintaining software products in order to prevent or correct defects and their causes, and to respond to new requirements and environmental changes [1]. The quality of the software design has a considerable impact on software maintainability [2]. Predicting software maintainability during the software design phase can reduce much of the maintenance costs and efforts, and improve software maintenance. While a number of research studies performed were based on measures taken after the coding phase, the cohesion metric we developed has an advantage of measuring cohesion in an earlier phase, the design phase. Another advantage of this metric is that it has been developed based on well-known and well-accepted package cohesion principles [3]. Further, if there is a relationship between our metric and software maintainability, then we will potentially establish a relationship between these principles and software maintainability.

This paper investigates the relationship between package cohesion, using the proposed metric CH, and software maintenance efforts. For this purpose, the package cohesion metric has been developed, based on a solid theory of the package design principles [3]. A number of experiments and statistical analyses have been designed and performed to investigate this relationship.

Looking carefully to the existing studies, some studies were conducted using a cohesion metric on the class level. Others were not validated or only validated theoretically without any empirical validation of the relationship with software maintenance. Some studies [4] used a subjective expert's surveys. Some related experimental studies [5-10] were performed to investigate some aspects of software maintenance, such as defect density or fault-proneness, but they don't consider other types of maintenance, such as adaptive maintenance. Some studies [11][12] did not rely on the reported maintenance history of the studied software systems. The drawback in such studies is that the maintenance data collected for the experimental studies does not represent the actual maintenance data. Some studies, such as [13][14], have relatively a small sample size of the experimental study, which makes the results hard to be generalized.

In contrast, we found that our study is unique in several different ways. It proposes a cohesion metric on a package level based on the well-known package cohesion principles, both theoretically and experimentally validated, uses actual maintenance data history of software, uses objective data instead of subjective ones, and considers all types of maintenance activities. To the best of our knowledge, there is no study that has investigated the relationship between package level cohesion and software maintainability, which makes this research original and vital in this matter.

The rest of this paper is organized as follows: The related studies are briefly introduced in Section II. Section III presents an overview of the studied package cohesion metrics. Section IV details the empirical study. Section V investigates and discusses the correlation between package cohesion and maintenance effort. Finally, Section VI concludes the paper with future works.

II. Related Work

Many researchers and practitioners proposed software metrics in relation to software maintainability and its characteristics. While some of them were theoretically validated, only a few were empirically validated. Several research studies were conducted to investigate the relationship between class-level cohesion and software maintainability. One of the early investigation studies was by Li and Henry [13] to investigate the validity of object-oriented metrics in predicting software maintenance efforts. The study tested if there is a strong relationship between object-oriented software metrics and maintenance efforts. LCOM, a cohesion metric developed by Chidamber and Kemerer [15], was among ten software metrics that were investigated. The results of the statistical analysis performed on two software systems showed that there is a strong relationship between the studied software metrics and maintenance efforts. Briand et al. [16] proposed cohesion and coupling measures based on object-oriented design principles to evaluate software maintainability. However, this approach was not validated. Briand et al. [17] defined a ratio-scale metric for cohesion to predict the error-proneness in the software design. The results of the experiments proved that software metrics can predict software error-proneness. Dagpinar and Jahnke [14] provided empirical evidence that software metrics can effectively be used to predict software maintainability. However, they found that Bieman and Kang's Loose Class Cohesion (LCC) [18], metric was not a significant predictor for class maintainability. Basili et al. [19] were concerned about fault detection and the fault prone-ness part of maintenance. They showed by their experiments' results that the Chidamber and Kemerer's metrics [15] are, individually, good indicators for faulty modules. This was supported by Gyimothy et al. [122] where a validation of the ability of the LCOM metric as a good indicator of software fault-proneness was indicated. The study was conducted on open source software, Mozilla. Koru et al. [20] showed that there is a correlation between the number of bugs and size. Al Dallal [119] empirically investigated the relationship between a number of internal class quality attributes (size, cohesion, and coupling) and class maintainability. Prediction models, based on statistical techniques, were constructed and validated to estimate the class maintainability. The results showed that internal attributes (size, cohesion, and coupling) have an impact on class maintainability. The higher the cohesion is, the higher the class maintainability is.

III. Package Cohesion Metrics

A. The proposed metric (CH)

In our previous work [22], which is motivated by Martin's package cohesion principles [3], we proposed two different cohesion metrics to measure two different cohesion concepts or types based on Martin's package cohesion principles in [3]. The first cohesion type, Common Reuse (CR), includes the factors that help in assessing CR cohesion. Similarly, the second cohesion type, Common Closure (CC), includes the factors that help in assessing CC cohesion. After each type of cohesion is measured by itself, the two values of CR and CC may be combined to one unified value of package cohesion, while still recognizing the two types.

The CR metric measures cohesion based only on the common reuse factors of the package. The elements of a package have different degrees of reachability. Reachability of a class in a package is the number of classes in the same package that can be reached directly or indirectly. The CR metric is defined as follows:

"Let $c \in C$, and suppose there is an incoming relation to c from a class in a different package. Then c is called an in-interface class. The cardinality of the intersection of the hub sets of all the in-interface classes in C divided by the number of classes in C is the CR of P ".

$$CR = |\cap_{\text{In-interface class hub sets}}| \,/\, |C| \qquad (1)$$

where

$$\text{Hubness}(c) = \{d \in C: \text{if there is a path } c \rightarrow d\}$$

C: set of classes in package *P*

c and d: classes in C

The CC metric considers the package dependencies on other packages as well as the internal dependencies between classes of the package. The classes of the package should depend on the same set of packages and, thus, they will have the same reasons for a change. The CC metric is defined as follows:

"The cardinality of the intersection of the reachable sets divided by the cardinality of the union of the sets represents the CC of P ".

$$CC = \left(|\cap_{\text{Reachable Package sets}}| \,/\, |\cup_{\text{Reachable Package sets}}| \right) \qquad (2)$$

The combined cohesion *CH* is defined as follows:

$$CH = \frac{\sqrt{2} - D}{\sqrt{2}} \qquad (3)$$

$$D = \sqrt{(1 - CR)^2 + (1 - CC)^2} \qquad (4)$$

B. Martin's metric (H)

Martin proposed a rational cohesion metric for the package,

$$H=(R+1)/N \qquad (5)$$

Where R: number of relationships between classes in the package

N: number of classes in the package

Although Martin's cohesion principles [3] are well known and well accepted, H metric doesn't conform to them. H measures the ratio of the relationships between classes of the package. This simple concept doesn't measure the common reuse or the common closure of the package, but rather, in its best situation, it may measure the classes' extent of being connected. The H metric depends on the number of relations rather than how these relations are designed. In this case, a well-designed package and a badly designed package could have the same cohesion value. In our previous work [22], further discussions are presented.

IV. Descriptive Statistics

This empirical study is based on four open-source Java software systems used to investigate the relation of package cohesion measure to software maintainability. This section provides descriptions about the studied software systems and the maintenance data collection. Two package cohesion metrics are included in this study: Martin's cohesion metric (H) and the proposed package cohesion metric (CH), which is developed based on Martin's package cohesion principles [3].

A. The software systems

Four open-source Java software systems were involved in the empirical study. All the four systems were selected based on the following criteria to allow results' generality; they had: (1) to be implemented using the Java programming language, (2) to have maintenance repositories available, namely Apache Subversion (SVN), (3) to have sufficient number of versions for each system that have been maintained, (4) to be organized using packages, (5) to have different sizes ranging from very large to small systems in terms of number of packages and number of classes, (6) to be from different domains, and (7) to have positive reviews and to be mature. We expect these criteria will allow the generalization of the results obtained from the study. The first system, Camel [23], is a rule-based and mediation engine to configure routing and mediation rules. The second system, Tomcat [24], is an open source webserver developed to implement Javaservlet and Java Server pages (JSP). Apache Tomcat is developed by the Apache Software Foundation. It has been developed and released under Apache License version 2. The third system, JHotDraw [25], is a Java GUI framework for technical and structured graphics. The fourth system, JEdit [26], is an open source Java text editor for programmers. It is licensed by GPL General Public License version 2.0. Table I provides details of the maintenance history; and Table II provides details about the studied systems.

TABLE I. MAINTENANCE HISTORY

	Base Release	End Release	History Studied
Camel	2.0.0	2.2.0	Aug/24/09 – Feb/6/10
Tomcat	7.0.6	7.0.22	Jan/14/11 – Oct/1/11
JHotDraw	7.5	7.6	July/29/10 – Jan/9/11
JEdit	4.5.0	5.1.0	Jan/31/12 – July/28/13

TABLE II. THE STUDIED SYSTEMS

	#LOC	#Methods	#Classes	#Packages	#Revised-Packages
Camel	143732	17369	5111	264	179
Tomcat	170461	15372	1725	113	62
JHotDraw	77194	7122	1026	65	65
JEdit	111861	7386	1238	35	23

B. Maintenance data

The source of the maintenance data for this study is the Version Control System (VCS), subversion (SVN), which is publicly available. The public can view the history of maintenance activities that have been made on the software system using SVN client. Each log entry in the repository log has a revision number, date and time, and short message that explains the maintenance activity. We considered all types of maintenance activities: perfective, adaptive, corrective, and preventive. We don't differentiate between different maintenance activities.

For this empirical study, as suggested by Al Dallal [21], we considered two package maintenance measures: the number of revisions (#Revisions) in which the package has been involved, and the number of revised lines of code (RLOC) during the studied maintenance history. The number of revised lines of code RLOC is calculated as suggested by Li and Henry [13], where a line added or deleted is considered one revised line, and a line modified is considered two revised lines, one deletion and one addition. We consider these two measures for two reasons. First, the number of revisions refers to the maintenance rate, while the number of RLOC is found to be correlated with maintenance cost [27][21] and maintenance effort measured in unit of time [28][21]. Packages with lower maintenance rates are better than those with higher rates because the code with more revisions becomes less organized, less understandable, and more fault-prone [29][21]. Second, these two measures are measurable using the freely available software maintenance history [21].

To collect maintenance data, we used the free software tool, TortoiseSVN [30], which is a subversion client developed to access the subversion (SVN) repositories. For each software system, the log of the SVN repository includes the following revision information: revision number, revision description, all the packages and classes affected by the revision, the previous and the current class versions, and the number of lines added, deleted, or modified. We had to create a list of all the packages and the classes within the package to relate each revision's information to the appropriate package. Then, revisions and revised lines of code were collected on package level. We considered different versions for each system, and collected the maintenance data reported during the entire maintenance period. Table III summarizes maintenance data for each system.

TABLE III. MAINTENANCE DATA

	#Revisions	Mean #Revisions	#RLOC	Mean #RLOC
Camel	1614	6.11	60688	229.87
Tomcat	636	5.63	22027	194.93
JHotDraw	354	5.45	21857	336.26
JEdit	323	9.23	9981	285.17

Two computer science PhD students were dedicated to collecting the maintenance data. The data was collected manually from the maintenance repositories. We have randomly checked the validity of the data collected. This process increased our confidence about the validity of the data collected.

For the purpose of a system's list of classes and list of packages, we have used the JHawk tool [31]. Then, each revision reported in the maintenance history was specified to the associated class along with the number of revised lines of code RLOC. Finally, maintenance data was collected on the package level.

C. Package cohesion data

Package cohesion data was gathered from two package cohesion metrics. The first metric is our proposed package cohesion metric, CH. The second metric is Martin's cohesion metric, H. These two metrics have been used to investigate the correlation between package cohesion and maintainability. For the purpose of data gathering, we have developed our Java tool to measure the CH package cohesion metric. The tool has been extended to calculate Martin's package cohesion metric, H. For each system, a list of all the packages, the number of classes in each package, and the associated cohesion values were generated.

V. Exploring the Correlation between Cohesion and Maintenance Effort

The correlation analysis aims to determine whether each individual package cohesion metric (CH and H) is significantly related to the maintenance measures (#Revisions and RLOC) of the package. For this purpose, we have performed Spearman's rank correlation due to the non-parametric nature of the metrics' data. We have used the well-known SPSS software for the correlation analysis of the empirical study. We have created and analyzed a correlation matrix for each software system in the study. Each correlation matrix has all the studied variables (cohesion and maintenance), a correlation coefficient (r), and significance level. For each pair of variables, r value can range between -1 and +1, where 1 represents a perfect positive correlation between the pair variables; -1 denotes a perfect negative correlation; and 0 indicates that there is no relationship between the variables. The magnitude of the coefficient determines the degree of the correlation.

Besides the strength of the correlation, the relationship between any pair of variables should be assessed for its significance as well. The significance is assessed by the p-value, which corresponds to the probability that the found correlation might be due to purely random effects. The smaller the p-level, the more significant is the relationship between variables [32]. The significance of the correlation in this empirical study was tested at a 95% confidence level (i.e., p-level $\leq$ 0.05). While the correlation can establish the relationship, it cannot establish a cause-effect relationship between the pair variables [32].

A. Hypotheses

Our objective is to assess to what extent is the package cohesion metric related to the maintenance effort of the software packages. The hypotheses of the empirical study are:

H_{01}: There is no significant correlation between package cohesion, CH, and the number of Revisions, #Revisions.

H_{02}: There is no significant correlation between package cohesion, CH, and the number of revised lines of code, RLOC.

H_{03}: There is no significant correlation between Martin's package cohesion, H, and the number of Revisions, #Revisions.

H_{04}: There is no significant correlation between Martin's package cohesion, H, and the number of revised lines of code, RLOC.

In this experiment, rejecting the null hypothesis indicates that there is a statistically significant relationship between the pair of variables (significance level $\alpha = 0.05$).

B. Statistical Analysis

The number of software revisions (#Revisions) and the number of revised lines of code (RLOC) on the package during the maintenance history assess software package maintainability. A lower number of package revisions and a smaller number of revised lines of code during the package maintenance history indicates less effort needed to maintain the software and thus, indicate high maintainability.

Table IV provides descriptive statistics (mean and standard deviation) for the variables used in analyzing software maintainability across the four systems, Camel, Tomcat, JHotDraw, and JEdit. We included Martin's package cohesion metric (H) in the list of variables for the purpose of comparison.

TABLE IV. Means and Standard Deviations of the Variables Used in the Maintainability Analysis

Variable	Camel N=264		Tomcat N=113		JHotDraw N=65		JEdit N=35	
	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
H	.636	.361	.817	.524	.705	.502	1.059	1.075
CH	.530	.388	.358	.374	.288	.317	.374	.417
#Classes	13.700	29.637	16.17	23.063	16.31	18.332	35.37	47.888
#Revisions	6.114	14.91	5.575	12.238	5.45	3.192	9.23	17.066
RLOC	229.879	732.318	194.69	511.186	336.26	401.748	285.17	573.679

C. Results and Discussion

A Spearman Rho correlation is the appropriate measure of a bivariate relationship when normality and linearity conditions for the Pearson's product moment correlation do not hold. For this study, the Spearman Rho correlation provides a measure of association between the proposed measure of package cohesion CH, the Martin's package cohesion metric H, package size (#Classes), and the two measures of package maintainability, the number of package revisions (#Revisions) and the number of revised lines of code (RLOC), within each of the four data sets. Table V provides the list of these correlations for the four sets of data.

TABLE V. SPEARMAN'S RHO CORRELATIONS FOR MAINTAINABILITY ANALYSIS

Data Set		H	CH	#Classes	#Revisions
	CH	.281**			
Camel	#Classes	-.350**	-.655**		
N=264	#Revisions	-.101	-.562**	.720**	
	RLOC	-.129*	-.533**	.702**	.962**
	CH	.169			
Tomcat	#Classes	-.069	-.736**		
N=113	#Revisions	-.010	-.545**	.686**	
	RLOC	-.007	-,521**	.663**	.792**
	CH	.157			
JHotDraw	#Classes	-.041	-.706**		
N=65	#Revisions	-.07	-.594**	.674**	
	RLOC	.098	-.631**	.769**	.792**
	CH	.468**			
JEdit	#Classes	-.205	-.709**		
N=35	#Revisions	-.024	-.650**	.754**	
	RLOC	-.008	-.623**	.711**	.983**

** Correlation is significant at the .001 level
* Correlation is significant at the .05 level

Table V reveals that the new proposed measure of package cohesion, CH, consistently has a negative large correlation with the two measures of package maintainability, number of package revisions (#Revisions) and the number of revised lines of code (RLOC), across all the four data sets. The correlation values between package cohesion CH and number of revisions (#Revisions) across the four data sets range from -0.545 (for the Tomcat system data set) to -0.650 (for the JEdit system data set). Similarly, the correlation values between package cohesion CH and the number of revised lines of code (RLOC) across the four data sets ranges from -0.521 (For the Tomcat system data set) to -0.631 (for the JHotDraw system data set). The statistically significant correlations confirm that the expectation of a highly cohesive software package requires less effort to maintain. That is high values of the proposed measure of package cohesion are associated with a lower number of its revisions and a lower number of revised lines of code.

In this study, the correlations between Martin's package cohesion metric H and the two package maintainability measures, number of package revisions (#Revisions), and the number of revised lines of code (RLOC) are not as strong as the ones with the newly proposed measure of package cohesion CH. These correlations are consistently weak and statistically insignificant across all the four data sets, except for the correlation with the revised lines of code (RLOC) for the Camel system's data. The value of the correlation is -.129, which relatively small yet statistically significant at an .05 level. The significance of the weak correlation might be justified by the large sample size of the Camel system data set. The correlation values between Martin's package cohesion H and number of revisions (#Revisions) across the four data sets range from -0.010 (for the Tomcat system data set) to -0.101 (for the Camel system data set). Similarly, correlation values between Martin's package cohesion H and the number of revised lines of code (RLOC) across the four data sets ranges from -0.007 (for the Tomcat system data set) to -0.129 (for the Camel system data set).

Table VI summarizes the results of the examined null hypotheses. In this experiment, rejecting the null hypothesis indicates that there is a statistically significant relationship between the pair of variables (significance level $\alpha = 0.05$).

TABLE VI. THE RESULTS OF THE NULL HYPOTHESES

	Camel	Tomcat	JHotDraw	JEdit
H_{01}	Rejected	Rejected	Rejected	Rejected
H_{02}	Rejected	Rejected	Rejected	Rejected
H_{03}	Accepted	Accepted	Accepted	Accepted
H_{04}	Rejected	Accepted	Accepted	Accepted

VI. CONCLUSION

This study investigated the relationship between the software internal attribute, package cohesion, and the software external attribute, package maintainability. We found that package cohesion, using our proposed metric (CH), is highly correlated with package maintainability, measured by number of revisions (#Revisions) and number of revised lines of code (RLOC). As high cohesion, the package is the easiest to be maintained. Such relationship is explained by the Spearman's ranking correlations involving data sets of four Java open-source software systems. This high correlation will lead us in future to perform regression analyses to predict package maintainability using package cohesion. Predicting software maintainability during the software design phase can reduce much of maintenance costs and efforts.

One strength of this study is the number of the studied systems and the stability of the correlation of CH across all experiments performed that allows us to draw optimistic conclusions about the possibility of using it as an indicator.

The experiments support the relationship between package cohesion and software maintainability, although it may behave differently based on a system's domain. So the results in this study should be viewed as indicative rather than conclusive.

The study only involved systems developed in Java, and the results could be different with systems developed in other object-oriented languages (such as C++).

References

[1] IEEE, IEEE standard glossary of software engineering terminology, IEEE Std 610.12-1990, Institute of Electrical and Electronics Engineering, 1990.

[2] Madhwaraj, K. G., and Chitra Babu. "An Empirical Investigation of the Influence of Object Oriented Design Quality Metrics on the Package Maintainability of Open Source Software." (2011).

[3] Martin, Robert Cecil. Agile software development: principles, patterns, and practices. Prentice Hall PTR, 2003.

[4] Muthanna, S.; Kontogiannis, K.; Ponnambalam, K. and Stacey, B., "A maintainability model for industrial software systems using design level metrics," *In Proceedings of 7th Working Conference on Reverse Engineering*, pages 248-256, 2000.

[5] Marcus, Andrian, Denys Poshyvanyk, and Rudolf Ferenc. "Using the conceptual cohesion of classes for fault prediction in object-oriented systems." *IEEE Transactions on Software Engineering,* 34.2 (2008): 287-300.

[6] [Al Dallal, Jehad. "Fault prediction and the discriminative powers of connectivity-based object-oriented class cohesion metrics." *Information and Software Technology* 54.4 (2012): 396-416.

[7] Al Dallal, Jehad, and Lionel C. Briand. "An object-oriented high-level design-based class cohesion metric." *Information and software technology* 52.12 (2010): 1346-1361.

[8] Briand, Lionel C., et al. "Predicting fault-prone classes with design measures in object-oriented systems." *The Ninth International Symposium on Software Reliability Engineering Proceedings*, 1998. IEEE, 1998.

[9] Gyimothy, Tibor, Rudolf Ferenc, and Istvan Siket. "Empirical validation of object-oriented metrics on open source software for fault prediction." *IEEE Transactions on Software Engineering*, 31.10 (2005): 897-910.

[10] Olague, Hector M., et al. "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes." *IEEE Transactions on Software Engineering*, 33.6 (2007): 402-419.

[11] Kabaili, Hind, Rudolf K. Keller, and Francois Lustman. "Cohesion as changeability indicator in object-oriented systems." *Fifth European Conference on Software Maintenance and Reengineering*, 2001. IEEE, 2001.

[12] Ajrnal Chaumun, M., et al. "A change impact model for changeability assessment in object-oriented software systems." *Proceedings of the Third European Conference on Software Maintenance and Reengineering*, 1999. IEEE, 1999.

[13] Li, Wei, and Sallie Henry. "Object-oriented metrics that predict maintainability." *Journal of systems and software*, 23.2 (1993): 111-122.

[14] M. Dagpinar and J. Jahnke, "Predicting Maintainability with OO Metrics – An Empirical Comparison", *10th Working Conference on Reverse Engineering Proc* (WCRE'03), 13-17 Nov, 2003, pp 155-164, 2003.

[15] Chidamber, Shyam R., and Chris F. Kemerer. Towards a metrics suite for object oriented design. Vol. 26. No. 11. ACM, 1991.

[16] Briand, Lionel C., Sandro Morasca, and Victor R. Basili. "Measuring and assessing maintainability at the end of high level design." *Conference on Software Maintenance Proceedings, 1993*. CSM-93, IEEE, 1993.

[17] Briand, Lionel, Sandro Morasca, and Victor R. Basili. "Defining and validating high-level design metrics." (1994).

[18] Bieman, James M., and Byung-Kyoo Kang. "Cohesion and reuse in an object-oriented system." ACM SIGSOFT Software Engineering Notes. Vol. 20. No. SI. ACM, 1995.

[19] Basili, Victor R., Lionel C. Briand, and Walcelio L. Melo. "A validation of object-oriented design metrics as quality indicators." *IEEE Transactions on Software Engineering*, 22.10 (1996): 751-761.

[20] Koru, A. Gunes, Dongsong Zhang, and Hongfang Liu. "Modeling the effect of size on defect proneness for open-source software." *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*. IEEE Computer Society, 2007.

[21] Al Dallal, Jehad. "Object-oriented class maintainability prediction using internal quality attributes." *Information and Software Technology* 55.11 (2013): 2028-2048.

[22] W. Albattah and A. Melton, "Package cohesion classification", in: *5th IEEE International Conference on Software Engineering and Service Science* (ICSESS), 2014, IEEE, 2014, (pp. 1–8).

[23] http://camel.apache.org (accessed March 2014)

[24] http://tomcat.apache.org (accessed March 2014)

[25] http://www.jhotdraw.org (accessed March 2014)

[26] http://www.jedit.org (accessed March 2014)

[27] Granja-Alvarez, Juan Carlos, and Manuel José Barranco-García. "A method for estimating maintenance cost in a software project: a case study." *Journal of Software Maintenance* 9.3 (1997): 161-175.

[28] Hayes, Jane Huffman, Sandip C. Patel, and Liming Zhao. "A metrics-based software maintenance effort model." *15th European Conference on Software Maintenance and Reengineering*. IEEE Computer Society, 2004.

[29] K. Erdil, E. Finn, K. Keating, J. Meattle, S. Park, D. Yoon, Software maintenance as part of the software life cycle, Comp180: Software Engineering Project, Department of Computer Science, Tufits University, 2003.

[30] http://tortoisesvn.net (Accessed March 2014)

[31] http://www.virtualmachinery.com/jhawkprod.htm (accessed Dec 2013)

[32] Gupta, Varun, and Jitender Kumar Chhabra. "Package level cohesion measurement in object-oriented software." Journal of the Brazilian Computer Society 18.3 (2012): 251-266.