Predicting Costly Maintenance Packages Using Package Cohesion

Waleed Albattah*
Information Technology Department, Qassim University, Qassim, KSA.

* Corresponding author. email: w.albattah@qu.edu.sa
Manuscript submitted August 31, 2018; accepted October 10, 2018.
doi: 10.17706/jsw.13.10.547-558

Abstract: Poorly designed software systems leads to difficult and costly maintenance activities. Object-oriented software metrics try to reflect the quality of the system design. Package cohesion metrics are found to be related to software maintainability. In our previous work, we developed a new package cohesion metric based on Martin's well-known and well-accepted package design principles. This metric was found to have a correlation with software maintainability and software testability. In this paper, we continue this empirical investigation by predicting costly maintenance packages using package cohesion. The prediction models created by logistic regression analysis show that costly packages can be predicted using package cohesion as early as the design phase of the software development lifecycle. This early prediction helps software engineers to discover any weak designs of the system and then these packages can be re-designed or at least carefully tested and documented, which contribute to considerable reductions in maintenance costs. The results demonstrate that the new package cohesion metric is promising and can help in locating low maintainable software packages.

Key words: Cohesion, maintainability, maintenance, measurement, metric, package, software.

1. Introduction

Software maintenance is the last and the longest phase in the software development lifecycle. It aims to keep a software product up to date and free of faults and defects. However, this phase depends on the early phases of the software development lifecycle, which makes it a very critical phase, since it spans starting from the delivery of the software product until it becomes outdated or obsolete. Although there are different definitions of software maintenance in the literature, all definitions concentrate on the changes applied to the software after its first installation. IEEE Standard [9] defined software maintenance as the "modification of a software product after delivery to correct faults, to improve the performance or other attributes, or to adapt the product to a modified environment." During this long phase, different software maintenance activities (including predictive, preventive, adaptive, and corrective changes) are conducted [2], [3]. Software maintenance has the largest share of the total cost of software [4], and developers spend at least half of their time analyzing and understanding software [5]. Unfortunately, although this phase is costly and a very challenging phase, it is poorly managed according to [10]. Therefore, designing a well maintainable software product is a very important goal. Giving that software maintainability is "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment" [1], developing good predictors for the software maintenance effort in the early phases of software, namely the design phase, can avoid much of the costs and efforts spent in every maintenance activity. The main goal of this research is to provide software developers with an evaluation of design alternatives at an early phase, the design phase, or even at the

maintenance phase in order to meet maintainability objectives. The research tries to present a prediction model for predicting costly maintenance packages during the design phase by using package cohesion and package size metrics.

Many research works have studied software design to improve the quality of open source software products. However, the measurement of software maintainability during an early phase, i.e. the design phase, can reduce maintenance cost and effort. During the design phase, one successful approach to identify costly maintenance packages is to create prediction models using object-oriented design metrics, such as cohesion and size metrics. These prediction models can be built and verified using real maintenance historical data. Prediction models can help to identify packages with potentially high maintenance cost and effort during the design phase. This early prediction can help developers take another design option to avoid unnecessary potential refactoring or high expected maintenance costs and efforts.

In object-oriented, design principles provide a roadmap to design maintainable software products. R. C. Martin [6] presented design principles to improve software design, which became well-known and well-accepted in the field of software design. Martin believes that following his design principles could improve software maintainability and quality. The current research has been established based on three package cohesion principles. Our previous work [11], [12] presented the new package cohesion metric that has been created based on Martin's package cohesion principles. This new metric has also been empirically validated [13], [14]. Package cohesion is meant to be the coherence of a package among its classes that should be closely related. Cohesion is an internal attribute of software quality that affects its maintainability. Following Martin's design principles [6], high cohesion is a goal to achieve software maintainability and promote its reusability [7], [8]. According to design principles [6], package design should maximize its cohesion and minimize its coupling with other packages in the software system. Our cohesion metric considers the factors that can affect software maintainability, and it has shown some encouraging results about the correlation between cohesion on one side and maintainability [13] and testability [14] on the other side. The current work is an extension to these previous studies in investigating the ability of our package cohesion metric along with a package size metric in predicting the maintenance of costly software packages.

The rest of this paper is organized as follows. In Section 2, we present the background of the topic, including internal and external attributes, related work, and our proposed package cohesion metric. Section 3 describes the design of the conducted study, while Section 4 reports and discusses its results. We discuss future work and close with conclusions in Section 5.

2. Background

Design level metrics have gained their importance as early potential predictors of the significant impact of the quality of software design on its maintainability [26]. Early prediction of software maintainability can reduce much of the maintenance costs and efforts. We hope that our cohesion metric helps to predict a costly maintenance package as early as the design phase which leads to cost reductions as well as to maintenance improvements. Although some research studies performed to predict software maintainability were based on measures taken after the coding phase, our cohesion metric has an advantage of measuring cohesion in an earlier phase, the design phase.

While there is no one single predictive measure for software maintainability [17], cohesion as one factor, which is measured by the proposed cohesion metric (CH), in this research study, can be counted as a predictor for software maintainability.

This paper presents a prediction model for costly maintenance packages at an early phase, the design phase, using a package cohesion metric developed based on a solid theory of the package design principles

[6] to predict software maintainability. The next sections presents an overview of the cohesion metric and related work.

2.1. Internal and External Attributes

Software product quality has internal and external attributes [19]. External attributes, such as maintainability, depends on different factors such as software environment, package design, the experience of the maintenance team, and the age of the system [18], [20]. As the software system get older, it is more likely to require more maintenance efforts because its size is likely to increase. As the size is increased, the software is more likely to be less organized and less understandable [20]. Although the package external attributes, such as maintainability, cannot be known before it has been maintained, the package's external attributes can be predicted using package internal attributes, such as cohesion, as indicators for the external ones.

In this empirical study, we followed the approach that has been utilized in different research studies in the literature, such as Al Dallal [20], Morasca [21], and Li and Henry [22]. We consider package cohesion as the internal attribute to predict the external attribute, package maintainability, using probabilities and a probabilistic model. Using a probabilistic approach has two advantages [21]. First, it is a theoretically well-defined and well-studied mathematical concept. Second, from the application point of view, probabilities are practically and commonly used in the field.

For the package internal attribute, package cohesion is used as an independent variable to predict the package external attribute, package maintainability. We include another package internal attribute, which is the package size (i.e., the number of classes in the package), to control for its effect and improve model prediction. The prediction process requires the actual data collection of maintenance from the actual software histories. We use three measures for package maintainability. The first measure is the Revised Package (RP), to predict packages likely to be involved in at least one revision. The second measure is Frequently Revised Package (FRP), to predict packages likely to be frequently revised. The third measure is the Costly Package (CP), to predict packages likely to show a substantial amount of Revised Lines Of Code (RLOC), i.e., added, deleted, or modified. These three measures will indicate the package maintainability.

2.2. Related Work

This section presents some related work on the measurement of software maintainability. Most of the studies in the literature provide maintainability indicators on a class level cohesion. Some of these studies theoretically or empirically studied the relationships between internal attributes, e.g., cohesion, and maintainability.

The literature also has several software metrics to evaluate internal quality properties, such as cohesion. Researchers and practitioners have proposed a number of cohesion metrics on a class level in object-oriented systems and related them to software maintainability and its qualities [36]. Whereas some of these metrics were hypothetically approved, just a couple of them were empirically validated. Li and Henry [22] made an investigation of the validity of objected-oriented metrics in predicting the efforts of software maintenance. This exploration tested whether there's a reliable connection between efforts in maintaining a software product and object-oriented software metrics. By measuring the quality of software through the use of Martin's design metrics [6], where cohesion metric H is one of the metrics, and applying it on software design, Atole and Kale [23] concluded that the design metric was able to make a software design quality evaluation. Dagpinar and Jahnke [24] confirmed that software metrics could effectively predict software maintainability. But they found that Bienman and Kang's loose class cohesion metrics (LCC) [25], did not significantly predict the class maintainability. Chidamber and Kemerer [26] proposed LCOM, a software metric that gives the results of a statistical analysis which is conducted on two software systems

and demonstrated that there was a substantial connection between the studied software metrics and efforts of maintaining the software. Another proposed cohesion metric based on design principles to assess software maintainability was conducted by Briand et al. [30]. They continued their research later [31] by defining a ratio-scale metric for cohesion for prediction of proneness to errors in the design of the software. Though the results of the experiment demonstrated that software metrics could predict how a software product is prone to errors, this approach was not validated. By precisely examining the connection between various internal class quality properties (cohesion, size, and coupling) and class maintainability, Al Dallal [20] proved that the higher the cohesion, the higher the maintainability of the class. The study proposed maintainability prediction models using statistical techniques. The study found that these models are reliable in evaluating the class maintainability. The results demonstrated that internal properties, for example cohesion, size, and coupling can affect maintainability of the class.

Carefully examining the current research, an investigation was done utilizing a cohesion metric on the class level, while others were not approved or just approved hypothetically with no specific approval of the association with software maintenance. some investigations did not depend on the reported maintenance history of the researched software systems. The disadvantage in such experimentation is that the support information gathered for the exploratory examinations does not present the accurate maintenance information.

some of the studies did not explore the capability of software measures in the prediction of software maintainability. That means the study does not include whether the internal property has an impact or not on the external property. A few investigations have a relatively small example size of the experimental study, which makes the outcomes challenging to be summed up. Other investigations are restricted to size and complexity properties and do not consider other imperative ones, such as coupling and cohesion.

Conversely, our investigation is distinct in a few different ways. It proposes a cohesion metric on a package level, both hypothetically and empirically approved, and utilizes real data from the systems' repositories. In our insight, this study will fill the gap in exploring the connection between package level cohesion and the used software maintenance measures, which makes this exploration unique and essential on this issue.

2.3. Package Cohesion Metric

In our previous work [11]-[16], we deeply discussed Martin's package cohesion principles [6]. Based on these principles [6], two separate cohesion metrics were created [11] to measure two parts of package cohesion, Common Reuse (CR) and Common Closure (CC) based on package cohesion principles. While CR measures the package in following Common Reuse Principle, CC measures the package in following Common Closure principle. Each metric is calculated separately, and then can be combined together to represent the overall cohesion value of the package. 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."

$$CR = |\cap_{\text{In-interface class hub sets}}| \ / \ |C|$$

where

Hubness(a) = {b $\in$ C: if there is a path a -> b}

C: set of classes in package P

a and b: classes in C

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."

$$CC = (|\cap \text{Reachable Package sets}| / |\cup \text{Reachable Package sets}|)$$

The combined cohesion CH is defined as follows:

$$\text{CH} = \frac{\sqrt{2} - D}{\sqrt{2}}$$
$$D = \sqrt{(1 - CR)^2 + (1 - CC)^2}$$

3. Descriptive Statistics

The experiments of this study is performed on some open-source software systems because of the availability of the maintenance data for such systems. The main goal for the experimental study is to explore how package cohesion can be used to predict costly software maintenance packages. This section describes some details about the software systems and the maintenance data collection. The package cohesion metric (CH), presented in Section 2 is included in the experiments as the independent variable.

3.1. The Software Systems

In the experimental study, four Java software systems were included, based on a set of criteria. We believe these criteria will allow for the results to be generalized. All the four systems had: (1) to be developed using Java, (2) maintenance repositories publicly available, (3) different versions. Table 1 and Table 2 provide some details about these systems in the experiments.

Table 1. History Studied

	Base Release	End Release	History Studied
System 1	2.0.0	2.2.0	Aug/09 – Feb/10
System 2	7.0.6	7.0.22	Jan/11 – Oct/11
System 3	7.5	7.6	July/10 – Jan/11
System 4	4.5.0	5.1.0	Jan/12 – July/13

Table 2. Details of the Systems

	#LOC	#Classes	#Packages	#Revised-Packages
System 1	143732	5111	264	179
System 2	170461	1725	113	62
System 3	77194	1026	65	65
System 4	111861	1238	35	23

3.2. Maintenance Data

Similar to studies in the field, subversion (SVN), which is available online for public use is the maintenance data source for this study. The maintenance activities of open source software systems can be viewed using an SVN. Each entry in the repository log in the system maintenance history has a number and a note that describes the maintenance action. This study considered all types of maintenance activities. We don't distinguish between perfective, adaptive, corrective, or preventive activities because we believe all of them require maintenance efforts and consequently maintenance costs.

Following the approach suggested by Al Dallal [20], we considered three package maintenance measures: The first measure is the Revised Package (RP), to predict packages likely to be involved in at least one revision. The second measure is Frequently Revised Package (FRP), to predict packages likely to be frequently revised. The third measure is the Costly Package (CP), to predict packages likely to show a

substantial amount of Revised Lines Of Code (RLOC), i.e., added, deleted, or modified. These measures can represent the maintenance effort and costs [20], [33], [32]. As much as the packages is maintained it becomes more fault-prone and less consistent [20], [10]. Additionally, all the three measures can be obtained from the publicly available maintenance data [20]. We believe that these three measures will indicate the package maintainability.

Two main software tools were used to collect maintenance data, TortoiseSVN [41] and JHawk tool [42]. TortoiseSVN provides the access to software maintenance repositories that include the details of each revision or maintenance action. The maintenance data were collected on the package level. JHawk tool provides the list of packages and the classes within packages for each software system. All the maintenance data was collected manually and randomly checked for the purpose of the data validity to increase the confidence about the data collected. Table 3 presents an overview of the collected data.

Table 3. Maintenance Data

	#Revisions	Mean #Revisions	#RLOC	Mean #RLOC
System 1	1614	6.11	60688	229.87
System 2	636	5.63	22027	194.93
System 3	354	5.45	21857	336.26
System 4	323	9.23	9981	285.17

3.3. Independent Variables

Package cohesion data was gathered using the package cohesion metric (CH) already presented in Section 2. The metric has been used to construct the maintainability prediction model. For the purpose of cohesion data, we have developed our Java tool to measure the CH package cohesion metric. For each system, a list of all the packages, the number of classes in each package, and the associated cohesion values were generated. We consider package cohesion (CH) and package size (# Classes) as the independent variables for this study.

4. Empirical Analysis and Results

In this section, we present another empirical study we performed that has been motivated by our previous works in investigating how package cohesion is related to package maintainability [13] and package testability [14]. Using logistic regression analysis, the study investigates the relationship between package cohesion and maintainability in terms of maintenance effort. We expect that a package that has low cohesion will likely require a high maintenance effort. We have used the logistic regression analysis, which is based on maximum likelihood estimation [17], to analyze the relationship between package cohesion and maintainability. Logistic Regression is a statistical modelling method used to predict a binary dependent variable [27], [39]. It measures the relationship between one or more independent variables and one dependent categorical variable, which can take only two values [17]. This kind of regression analysis is widely applied to predict the software external quality attributes, such as maintainability [20], [38], fault proneness [34], [35], [37], reusability [29], and testability [28], [39], [40].

In this study, we use logistic regression to predict the dependent variable (maintenance effort) from independent variable(s) (package cohesion and package size). Both univariate and multivariate logistic regression analysis has been performed. The univariate analyses are performed to investigate the individual effect of package cohesion on maintainability. The multivariate analyses are performed to evaluate the combined effect of package cohesion and package size on maintainability. The general univariate logistic regression model, in terms of the risk of the predicted variable, p, is given by:

$$p = \frac{e^{\beta 0+\beta 1x}}{1+e^{\beta 0+\beta 1x}}$$

The general multivariate logistic regression model is given by the following equation:

$$p = \frac{e^{\beta 0+\beta 1x1+\cdots+\beta pxp}}{1+e^{\beta 0+\beta 1x1+\cdots+\beta pxp}}$$

Where p is the probability of finding a high maintenance effort package. xi, i=1,2,..., n, are the independent variables and β, i=1,2,...,n are the estimated regression coefficients for each independent variable xi. The strength of the impact of the independent variable is determined by the absolute value of the coefficient. For each independent variable, xi, the p-value is computed and assessed by the (α = 0.05) significance level [39]. R2 is the proportion of the total variance in the dependent variable that is explained by the model. As high as the R2 value just as high is the impact of the independent variables on the dependent ones, and as accurate is the model [17], [39].

4.1. Dependent Variables

Three binary dependent variables in this study are used to measure the maintainability of packages. We consider maintainability from the maintenance effort point of view. As presented by Al Dallal [20], we consider predicting three practical problems: (1) to predict packages likely to be involved in at least one revision, (2) to predict packages likely to be frequently revised, and (3) to predict packages likely to show a substantial amount of RLOC. Predicting these three types of packages will help developers during the software design phase to focus on packages that are more likely to be frequently revised and those are more likely to be costly in terms of RLOC [20]. Based on the above practical problems, three dependent variables are defined for the logistic regression analysis as follows:

- Revised Package (RP) is the package that has been involved in at least one revision during the maintenance history of the software. RP takes 1 value if it has been involved and takes 0 if it has not at all.
- Frequently Revised Package (FRP) is the package that has been involved in revisions more than the average number of revisions of all the packages in the system. If the number of revisions of the package is equal to or greater than the average, the FRP is equal to 1; otherwise, it equals 0.
- Costly Package (CP) is the package that has a number of RLOC equal to or greater than the average RLOC number among all the packages in the system. In such a case, CP takes the value of 1; otherwise, it takes the value of 0.

4.2. Statistical Analysis

Binary Logistic regression provides us with another look at the relationship between the new proposed measure of the package cohesiveness CH and package maintainability. For logistic regression, package maintainability assessed by three binary distributed variables. These variables are modified (dichotomized) versions of the same variables used in the regression analysis to assess package maintainability. This modification of the variables allows us to look at package maintainability with consideration to the inherited nature of dichotomy in the distributions of the original variables used to assess package maintainability. The first variable is revised package (RP) which takes a value of one if the package has been revised at least one time and a value of zero if the package has not been revised at all. The second variable is the frequently revised package (FRP), which takes a value of one if the number of revisions of the package is

equal to or more than the average number of revisions of all packages and zero if the number of revisions of a package is less than the average number of revisions across all the packages. Finally, the third variable of assessing package maintainability is a costly package (CP). This variable takes a value of one if the number of revised lines of code is equal to or more than the average number of the revised lines of code RLOC across all packages and a value of zero if the number of revised lines of code is less than the average number of revised lines of code across all packages. Table 4 presents the distributions of the three binary variables of package maintainability from the combined data sets of the four systems.

Table 4. Distributions of the Package Maintainability Measures RP, FRP and CP

Values	RP		FRP		CP	
	N	%	N	%	N	%
0	144	30.19	377	79.04	381	79.87
1	333	69.81	100	20.96	96	20.13

N=477

Table 5 presents descriptive statistics (means and standard deviations) for the three binary measures of maintainability, RP, FRP, CP, the proposed measure of package cohesion (CH) and package size (#Classes).

Table 5. Means and Standard Deviation for RP, FRP, CP, CH and Classes

Statistics	RP	FRP	CP	CH	#Classes
Mean	.698	.210	.201	.445	16.23
STD	.460	.408	.401	.389	29.154

N = 477

4.3. Results and Discussion

We set up the favourable outcome for any of the above three binary variables, revised package (RP), the frequency of package revisions (FRP) and costly package (CP) to be equal to one. Logistic regression helps us predict the probability of the favourable outcome in each of the above three variables of maintainability controlling for the package size (#Classes).

RP logistic regression model

Table 6 presents a summary of the results of the logistic regression for the revised package (RP) outcome variable. The proposed package cohesion measure (CH) and package size (#Classes) are the predictors of the status of the package as it has been revised versus the package which has not been revised. The logistic regression results indicate that the overall model was statistically reliable in distinguishing between revised and non-revised package (-2log Likelihood = 444.042, $\chi 2$ (2) = 140.240, p <.000). The model correctly classifies (Hit rate) 77.4 per cent of the cases. Table 6 presents the regression coefficients with their statistical significance, the odd ratios associated with the coefficient, and their 95% confidence intervals.

Table 6. Regression Coefficients for Predicting RP by CH and #Classes

	B	Wald	df	Sig.	Exp(B)	95% C.I. for EXP(B)	
						Lower	Upper
CH	-1.237	12.498	1	.000	.290	.146	.576
#Classes	.130	23.093	1	.000	1.139	1.080	1.201
Constant	.473	57.452	1	.147	1.604		

Wald test statistics for the proposed measure of package cohesion (CH) is 12.498, which is statistically significant at p=.000. There is little (.290) change (decrease) in the odd ratio of package revision (RP) for

one unit increase in the package cohesion. That is as package cohesion increases by one unit, the odds of the package being revised will drop by .290.

FRP logistic regression model

Table 7 presents the results of the logistic regression analysis for the frequency of package revisions (FRP) variable for package maintainability. Similar to the revised package variable (RP), the model uses package cohesion (CH) and package size (Classes) to predict the status of the package as being revised as many times as or more than the average number of revisions of all packages versus the package has been revised less than the average of number of revisions for all packages. Regression results indicate that the overall model was statistically reliable in distinguishing between the two categories of FRP, equal or more than the average number of revisions versus less than the average number of reversions for all packages, (-2log Likelihood = 309.635, $\chi 2$ (2) = 180.228, p <.000). The model correctly classifies (Hit rate) 85.7 per cent of the cases. Table 7 presents the regression coefficients with their statistical significance, the odd ratios associated with the coefficient, and 95% confidence intervals for the odd ratio.

Table 7. Regression Coefficients for predicting FRP by CH and #Classes

	B	Wald	df	Sig.	Exp(B)	95% C.I. for EXP(B)	
						Lower	Upper
CH	-3.63	21.623	1	.000	.027	.006	.122
#Classes	.052	31.911	1	.000	.105	1.035	1.073
Constant	-1.325	20.801	1	.000	.266		

Wald test statistics for the package cohesion (CH) variables, 21.623 is statistically significant at p=.000. Yet the change it created in the odd ratio of the package frequent revisions (FRP) is minimal (.027). That is, for one unit increase in the package cohesion (CH) variable and controlling for its size (#Classes), the odds of being in the category of the package as being revised as many times as or more than the average number of revisions of all packages decreases by .027.

CP logistic regression model

Table 8 presents the results of the logistic regression analysis for the costly package (CP) outcome variable. Similar to the previous two models, the newly proposed package cohesion (CH) and package size (Classes) are the predictors of the status of the package on the two categories of the costly package (CP) variable. That is either as being a package with a number of revised lines of code equal or more than the average number of revised lines of code from all packages versus the package has a number of revised lines of code that is less than the average number of revised lines of code from all packages. Regression results indicate that the overall model was statistically reliable in distinguishing between the two categories of the costly package (CP) variable, (-2log Likelihood = 333.515, $\chi 2$ (2) = 145.528, p <.000). The model correctly classifies (Hit rate) 85.1 percent of the cases. Table 8 presents the regression coefficients with their statistical significance, the odd ratios associated with the coefficient, and 95% confidence intervals for the odd ratio.

Table 8. Regression Coefficients for Predicting CP by CH and #Classes

	B	Wald	df	Sig.	Exp(B)	95% C.I. for EXP(B)	
						Lower	Upper
CH	-2.116	13.786	1	.000	.121	.039	.368
#Classes	.051	35.066	1	.000	1.052	1.035	1.070
Constant	-1.685	36.661	1	.000	.185		

Wald test statistics for the newly proposed package cohesion (CH) variables, 13.786 is statistically significant at p=.000. Package cohesion (CH) predicted change in the odd ratio of costly package variable (CP) is .121. That is, for one unit increase in package cohesion (CH) variable we expect that the odds of being in the category of a package with the number of revised lines of code equal or more than the average number of revised lines of code from all packages decreases by .121.

5. Conclusion and Future Work

This paper continued our previous works [11]-[16] and presented package maintainability prediction models using package cohesion. Prediction models can be potentially beneficial when software developers in the industrial field can utilize them. We believe the models presented in this paper can help a software development team as early as the design phase of the software development lifecycle. Developers can utilize these models to evaluate the package maintainability, and they can decide the most maintainable package design based on the predictions of the model.

One advantage of this study is the number of the studied systems and the relatively large sample used in the regression analyses, as well as using actual maintenance data. In previous work [12], [13], the cohesion metric (CH) was found to be a significant early predictor for software maintenance efforts; which encouraged us to conduct further investigation of software maintainability using the CH metric. The stability of the impact of CH across different statistic analyses performed either with relation to software maintainability [13] or software testability [14] allows us to draw optimistic conclusions about its use as an indicator. Furthermore, based on the obtained results, we can believe that following the package cohesion principles by Martin [6], which the (CH) cohesion metric was developed based on, can improve the software maintainability.

The results of the study support the capability of such an indicator, based on objective empirical studies, to predict software maintainability, although it may behave differently based on a system's domain. Thus, the results of this study should be viewed as indicative rather than conclusive.

Given that maintainability is affected by different factors, it would be very interesting to consider other metrics in future studies. Such metrics include but are not limited to coupling between packages and complexity, along with the metrics already used in this study, package cohesion, and package size, to predict the maintenance effort and testing effort. It would also be valuable to involve more different metrics for cohesion, coupling, and size; and compare them in regards to the prediction proficiency. In this study, we only included systems developed in the Java programming language, while the results could be different with systems developed in other object-oriented languages, which would be very interesting to investigate.

References

[1] Jay, F., & Mayer, R. (1990). IEEE standard glossary of software engineering terminology. *IEEE Std*.

[2] Mamone, S. (1994). The IEEE standard for software maintenance. *SIGSOFT SE Notes. 19(1)*, 75–76.

[3] Lientz, B. P., & Swanson, E. B. (1980). Software maintenance management: A study of the maintenance of computer application software in 487 data processing organizations.

[4] Ahn, Y., Suh, J., Kim, S., & Kim, H. (2003). The software maintenance project effort estimation model based on function points. *Journal of Software Maintenance Evolution: Research and Practice*, *15*, 71–85.

[5] Basili, V. R. (1997). Evolving and packaging reading technologies. *Journal of Systems and Software*.

[6] Martin, R. C. (2003). Agile software development: principles, patterns, and practices. *Prentice Hall PTR*.

[7] Biggerstaff, T. J., & Alan, J. P. (1989). *Software Reusability: Vol. 1, Concepts and Models*.

[8] Ponisio, L., & Oscar, N. (2006). Using contextual information to assess package cohesion. *Institute of Applied Mathematics and Computer Sciences*.

[9] IEEE Std 1219-1993: Standard for Software Maintenance, IEEE Computer Society, 1992.
[10] Dagpinar, M., & Jahnke, J. (2003). Predicting maintainability with OO metrics – An empirical comparison. *Proceedings of the 10th Working Conference on Reverse Engineering Proc*.
[11] Albattah, W., & Austin, M. (2014). Package cohesion classification. *Proceedings of the 2014 5th IEEE International Conference on Software Engineering and Service Science*.
[12] Albattah, W. (2014). Software maintainability and testability predictions using package cohesion. *Diss*.
[13] Albattah, W. (2017). An empirical investigation of the correlation between Package-level cohesion and maintenance effort. *International Journal of Advanced Computer Science and Applications*, *8(3)*.
[14] Albattah, W., & Austin, M. (2017). An empirical analysis of the relationship between cohesion and testing effort. *Journal of Software*, 671-682.
[15] Albattah, W., & Alsuhibany, S. (2015). Revisiting package-level cohesion approaches. *Proceedings of the Tenth International Conference on Software Engineering Advances*.
[16] Almugrin, S., Waleed, A., & Austin, M. (2016). Using indirect coupling metrics to predict package maintainability and testability. *Journal of Systems and Software*, *121*, 298-310.
[17] Hosmer, J., David, W., & Stanley, L. (2016). *Applied Logistic Regression*. John Wiley & Sons.
[18] Fenton, N. E., & Shari, L. (1998). Pfleeger. Software metrics: A rigorous and practical approach.
[19] Fenton, N. (1994). Software measurement: A necessary scientific basis. *IEEE Transactions on Software Engineering*.
[20] Al, D. J. (2013). Object-oriented class maintainability prediction using internal quality attributes. *Information and Software Technology*.
[21] Morasca, S. (2009). A probability-based approach for measuring external attributes of software artifacts. *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*.
[22] Wei, L., & Sallie, H. (1993). Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 111-122.
[23] Atole, C. S., & Kale, K. V. (2006). Assessment of package cohesion and coupling principles for predicting the quality of object oriented design. *Proceedings of the Ist International Conference on Digital Information Management*.
[24] Dagpinar, M., & Jahnke, J. (2003). Predicting maintainability with OO metrics – An empirical comparison. *Proceedings of the 10th Working Conference on Reverse Engineering Proc (WCRE'03)*.
[25] Bieman, J. M., & Byung-Kyoo, K. (1995). Cohesion and reuse in an object-oriented system. *ACM SIGSOFT Software Engineering Notes*.
[26] Madhwaraj, K. G., & Chitra, B. (2011). An empirical investigation of the influence of object oriented design quality metrics on the package maintainability of open source software.
[27] Basili, V. R., Lionel, C. B., & Walcelio, L. M. (1996). A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*.
[28] Badri, M., & Toure, F. (2012). Empirical analysis of object-oriented design metrics for predicting unit testing effort of classes. *Journal of Software Engineering and Applications*, *5(7)*, 513-526.
[29] Al, D. J., & Sandro, M. (2014). Predicting object-oriented class reuse-proneness using internal quality attributes. *Empirical Software Engineering*, 775-821.
[30] Briand, L. C., Sandro, M., & Victor, R. B. (1993). Measuring and assessing maintainability at the end of high level design. *Proceedings of the Conference on Software Maintenance Proceedings*.
[31] Briand, L., Sandro, M., & Victor, R. B. (1994). Defining and validating high-level design metrics.
[32] Granja-Alvarez, J. C., & Manuel, J. B. (1997). A method for estimating maintenance cost in a software project: a case study. *Journal of Software Maintenance*.

[33] Hayes, J. H., Sandip, C. P., & Liming, Z. (2004). A metrics-based software maintenance effort model. *Proceedings of the 15th European Conference on Software Maintenance and Reengineering*.

[34] Briand, L. C., John, D., Victor, P., & Wust, J. (1998). Predicting fault-prone classes with design measures in object-oriented systems. *Proceedings. The Ninth International Symposium on Software Reliability Engineering*.

[35] Dallal, A. J., & Lionel, C. B. (2010). An object-oriented high-level design-based class cohesion metric. *Information and Software Technology*, 1346-1361.

[36] Lee, Y., & Kai, H. C. (2000). Reusability and maintainability metrics for object-oriented software. *Proceedings of the 38th Annual on Southeast Regional Conference*.

[37] Marcus, A., Denys, P., & Rudolf, F. (2008). Using the conceptual cohesion of classes for fault prediction in object-oriented systems. *IEEE Transactions on Software Engineering*.

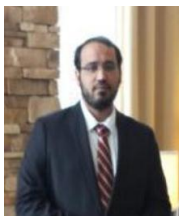
[38] Muthanna, S., Kontogiannis, K., Ponnambalam, K., & Stacey, B. (2000). A maintainability model for industrial software systems using design level metrics. *Proceedings of the 7th Working Conference on Reverse Engineering*.

[39] Badri, L., Mourad, B., & Fadel, T. (2011). An empirical analysis of lack of cohesion metrics for predicting testability of classes. *International Journal of Software Engineering and Its Applications*.

[40] Badri, M., & Touré, F. (2012). Evaluating the effect of control flow on the unit testing effort of classes: An empirical analysis. *Advances in Software Engineering*.

[41] Tortoise SVN. Retrieved August 30, from http://tortoisesvn.net

[42] JHawk - The Java metrics tool - Product Overview. Retrieved August 30, from http://www.virtualmachinery.com/jhawkprod.htm



Waleed Albattah received his Ph.D. from Kent State University, Ohio, USA. Dr. Albattah is a faculty member at the Information Technology Department, Qassim University, Saudi Arabia. His research interests are software engineering, software measurements, software design and agile software development, and software quality. Recently, he has been working in Big data and cloud computing security projects. He is the dean of College of Computers at Qassim University, and he is a member in the ACM Society SIGSOFT.