The Effects of Incorporating Special Methods into Cohesion Measurement on Class Instantiation Reuse-Proneness Prediction

Jehad Al Dallal
Department of Information Science
Kuwait University
P.O. Box 5969, Safat 13060, Kuwait
j.aldallal@ku.edu.kw

Abstract
The previously proposed class cohesion measures employ different approaches to assess the strength of the relations between the attributes and methods in a class. Access methods, constructors, and destructors are special types of methods with special characteristics that can falsely alter the class cohesion measurement. In this paper, we empirically explored the impact of considering special methods (SPs) on the cohesion measures' abilities to predict the classes that can be intensively reused via instantiation (IRI). We considered classes in the JHotDraw and Eclipse systems. For each class, we obtained cohesion results using 17 measures in four different scenarios of considering or ignoring SPs. We collected the instantiation reusability data and applied a statistical technique to build a prediction model using each measure in each considered scenario. We investigated the significance of the changes in the prediction results. Our results demonstrated that cohesion had a negative impact on class instantiation reuse-proneness and that SPs had significant impacts on cohesion values and the abilities of the cohesion measures to predict IRI classes. In practice, when applying cohesion measures to predict IRI classes, the results suggest that SPs must be included in cohesion measurement.

Keywords: object-oriented design, class quality, class cohesion, cohesion measure, special methods, instantiation reuse-proneness prediction.

1. Introduction

In object-oriented systems, a class is the basic unit of design and comprises attributes and methods. Several attributes for assessing class quality have been identified, and they are categorized as internal or external quality attributes 0. Internal quality attributes, e.g., size, cohesion, and coupling, are those that can be measured based only on class artifact knowledge, such as source code and design artifact knowledge. Conversely, external quality attributes, e.g., maintainability, testability, fault-proneness, and reuse-proneness, are those that indicate class quality based on factors that cannot be measured using only software artifact knowledge but rather other factors, such as the environment in which the system is deployed 0. Software practitioners are not interested in internal quality attributes, such as coupling and cohesion, unless these attributes are demonstrably related to external quality attributes, such as maintainability and reusability 0. For example, class cohesion is worth measuring only if it is believed to be or demonstrably related to such factors as the external attributes of the same artifact (e.g., class reuse-proneness). When proposing a measure to quantify an internal quality attribute, one must provide evidence

for the practical usefulness of the measure by, e.g., exploring its ability to predict one or more external quality attributes. If the internal quality attribute has several aspects and varieties, such as the types of methods and relations to be considered, the impact of these aspects and varieties on the application for which the internal quality attribute was determined useful must be studied. Otherwise, the internal quality attribute may not make the optimal contribution to the prediction model for the external quality attribute under consideration.

Class cohesion measurement aims at quantifying the degree of relatedness of the methods and attributes in a class [3]. A class can contain methods with different characteristics. For example, on one extreme, a constructor typically initializes most or all attributes in the class, whereas a destructor deinitializes most or all attributes. On the other extreme, an access method (i.e., a setter or getter method) references a single attribute. These methods, which are called SPs in this study, exhibit extreme cases with respect to the number or percentage of the accessed attributes in a class. Thus, the consideration of these methods in cohesion measurement potentially affects the obtained cohesion values and may alter the prediction results for the application of interest.

Considering or ignoring SPs is a variety whose impact on cohesion values and on applications of interest must be explored. The impact is expected to differ among measures because the existing cohesion measures follow different measurement approaches and apply different formulas. In practice, when considering a certain application of interest, e.g., predicting class reuse-proneness or maintainability, applying a specific scenario of considering or ignoring SPs, empirically determined to be the best, in cohesion measurement can lead to better prediction results. Considering SPs in cohesion measurement can have different and sometimes opposite effects on the prediction results for the different external quality attributes considered. It is thus incorrect to perform such a study based on one application and generalize the conclusions to all other applications. For example, considering SPs may negatively affect the fault-proneness prediction ability of a model that includes a certain cohesion measure, but it may also positively affect the prediction results of another external attribute using a prediction model that incorporates the same cohesion measure.

Several researchers have qualitatively addressed the problem of considering or ignoring SPs in cohesion measurement (e.g., [3, 4, 5, 6, 7]). To our knowledge, only Al Dallal [8] has empirically studied the effect of considering or ignoring SPs. In that study, a group of existing measures was selected and SPs were empirically investigated to determine their impact on the ability of the selected measures to predict class fault-proneness. In this paper, we use a different set of classes, consider a different analysis approach, and extend the previous study by considering another application of interest, namely, class reuse-proneness through inheritance. In this study, we account for predicting classes that are intensively reused via instantiation (IRI), and thus, we refer to the likelihood that a class can be intensively reused via instantiation as the class instantiation reuse-proneness. We investigate the relationship between the cohesion quality attribute and class instantiation reuse-proneness and explore the effect of considering or ignoring SPs on this application.

Thus, this study aims to guide software engineers and practitioners to the best scenario in which the cohesion measures can be applied when SPs are incorporated in prediction models for reusable classes.

The empirical study involves 17 cohesion measures that employ different cohesion measurement approaches and 2,079 classes of two widely considered Java systems in the research community, JHotDraw and Eclipse. For the SPs, we account for four possible scenarios: (1) ignoring all SPs, (2) considering constructors and ignoring access methods, (3) ignoring constructors and considering access methods, and (4) considering all SPs. In the empirical study, we do not account for destructors because they are not supported by Java, and we do not consider delegation methods because they are difficult to detect automatically [8].

We apply a statistical technique to build prediction models for class instantiation reuse-proneness using each cohesion measure applied in each of the four considered scenarios. Each prediction model is applied to determine whether each class is estimated to be reusable via instantiation. The results are used to empirically determine whether there is evidence of a relation between cohesion and instantiation reuse-proneness. In addition, the results are used to empirically decide which scenarios of considering or ignoring SPs are the best at predicting IRI classes. The results demonstrate that most of the considered cohesion measures are significant predictors of IRI classes and that cohesion has a negative impact on instantiation reuse-proneness. In addition, the results demonstrate that the cohesion values obtained and the corresponding instantiation reuse-proneness prediction results attained using most considered measures are significantly different among the four scenarios. These results indicate that it is essential for software practitioners to consider the problem of considering or ignoring SPs when incorporating cohesion measures in prediction models for external quality attributes. The results further indicate that choosing to consider SPs improves the instantiation reuse-proneness prediction abilities of the models that incorporate cohesion measures. We discuss the results and justify them intuitively.

The primary contributions of this paper are as follows.

1. It empirically investigates the abilities of cohesion measures to predict IRI classes.
2. It empirically explores the effects of considering or ignoring SPs in cohesion measurement on the capabilities of the 17 cohesion measures to predict IRI classes.

This paper is organized as follows. Section 2 reviews the literature on this topic. Section 3 reports and discusses the results of empirical studies that have investigated the impacts of cohesion on class instantiation reuse-proneness and explored the effect of considering or ignoring SPs on the capabilities of the measures to predict IRI classes. Section 4 lists validity threats to the empirical studies. Finally, Section 5 concludes the paper and discusses future work.

2. Related Work

In this section, we provide an overview of several existing class cohesion measures. We also review the existing research regarding the relation between cohesion and SPs. Finally, we provide an overview of existing research on the relation between cohesion and class reuse-proneness.

2.1. Class Cohesion Measures

Seventeen class cohesion measures are considered in this paper: LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, Coh, LSCC, CC, SCOM, TCC, LCC, DC_D, DC_I, CBMC, ICBMC, OL_n, and PCCC. The definitions of these measures are listed in Appendix A (Table A.1). All of the selected measures use information about attribute access in the methods within a class, and therefore, these measures are applicable during the implementation or low-level design phase. These measures were selected because they apply different measurement approaches; consequently, we can relate the results to the approaches employed by the measures. Al Dallal [8, 9] previously considered these measures in similar empirical studies, which allows us to compare our results with those previously obtained. The selected cohesion measures have been well studied from both theoretical and empirical perspectives [10, 11, 12, 13, 14, 15, 16].

2.2. Relations Between Cohesion and SPs

The original definitions of a few of the measures investigated here consider whether to consider or ignore SPs in cohesion measurement. Bieman and Kang [4] suggested ignoring constructors when applying TCC and LCC. Briand et al. [3] argued that considering constructors and destructors in cohesion measurement causes the values of LCOM1, LCOM2, LCOM3, and LCOM4 to artificially increase and that these SPs must be ignored as a result. Because the consideration of access method falsely affects the cohesion values obtained using TCC, LCC, LCOM1, LCOM2, and LCOM3, Briand et al. [3] also suggested either considering method invocations or ignoring access methods (when using TCC or LCC). When considering constructors, Etzkorn et al. [6] demonstrated by examples that the LCOM2 (LCOM3) value either decreases (increases) or remains the same. Without justifying or studying the impact of their choice on the DC_D and DC_I values, Badri and Badri [17] suggested accounting for all public methods, which typically include all SPs. Chae et al. [5] suggested ignoring SPs when applying CBMC because SPs do not contribute to class cohesiveness but potentially influence the cohesion measurement obtained. The ICBMC and OL_2 measures were originally defined to evaluate cohesion when ignoring SPs. Because constructors and destructors do not always access all attributes in a class, Fernandez and Pena [18] suggested considering constructors and destructors in cohesion measurement. In addition, they argued that access methods must be ignored because they always access a single attribute.

Al Dallal [8] empirically investigated the effect of considering or ignoring SPs in cohesion measurement on refactoring decisions and the measures' abilities to predict faulty classes. We extend Al Dallal's work by selecting a different set of classes, applying different statistical techniques, and investigating the impact of this problem on another application of interest, predicting class instantiation reuse-proneness.

2.3. Relation Between Cohesion and Reuse-Proneness

Bieman [19] identified two key class reuse mechanisms: instantiation (i.e., creating an object of a class) and inheritance. Each of these reuse mechanisms allows for the reuse of all members of a class except for private members. Class reuse-proneness is a key class quality attribute of interest for software practitioners [1]. Several researchers (e.g., [4, 20, 21]) have explored the relationship between class reuse-proneness as an external quality attribute and several internal class quality attributes, such as cohesion, coupling, and size. They theoretically or empirically investigated the use of these internal class quality attributes as predictors for the reuse-proneness of classes when they are developed. Bieman and Kang [4] empirically studied the ability of the TCC and LCC cohesion measures to predict inheritance and instantiation reuse-proneness. The results of the study indicated that class cohesion and inheritance reuse-proneness are inversely correlated. They did not detect any relationship between class cohesion and instantiation reuse-proneness. Barnard [20] empirical study results suggested that there is no relationship between LCOM1 and class reuse-proneness.

Al Dallal and Morasca [22] empirically investigated the relations between 21 class cohesion measures and both inheritance and instantiation reuse-proneness. They proposed statistical models to predict classes prone to be (1) reused at least once via instantiation, (2) reused at least once via instantiation, and (3) reused at least twice via instantiation (assuming that the creation of the first instance is a class *use* and the creation of the second instance is the actual first class *reuse*). The results demonstrated that the majority of the considered measures had positive impacts on instantiation reuse-proneness and negative impacts on inheritance reuse-proneness. However, it was found that the majority of the considered measures were only poor or fair at predicting inheritance and instantiation reuse-proneness.

In this paper, we consider a problem that was not considered by the previous studies, which is predicting the classes that are reused via instantiation not just once but rather intensively. We justify our choice for this reuse-proneness definition from a practical perspective and investigate whether considering SPs in cohesion measurement improves the abilities of the cohesion measures to predict class instantiation reuse-proneness as defined here. Because of space limitations, this paper does not consider other reuse mechanisms, such as class inheritance, although an analysis similar to that provided here is applicable to other reuse mechanisms.

3. Empirical Analysis of Reuse-Proneness Prediction

We empirically investigated the effects of considering SPs on the considered measures' abilities to predict instantiation reuse-proneness. The empirical study had the following two goals. The first goal was to empirically determine whether considering or ignoring SPs caused instantiation reuse-proneness prediction abilities to significantly change for the selected example measures. Consequently, the study empirically determined whether software engineers must include or exclude SPs when applying the considered cohesion measures to predict instantiation reuse-proneness. The second goal was to compare the

results with those of similar studies that have been performed to determine whether considering or ignoring SPs had the same effect on all applications of interest for software practitioners.

We considered four scenarios for considering or ignoring SPs: (1) ignoring all SPs, (2) considering constructors and ignoring access methods, (3) considering access methods and ignoring constructors, and (4) ignoring all SPs. This section provides descriptions of the selected systems and the data collection process, and it reports and discusses the empirical analysis results.

3.1. Selected Systems and Data Collection Process

We selected two open-source software systems, JHotDraw version 7.4.1 [23] and Eclipse version 1.2.0 [24]. We selected these software systems because they (1) are well known and widely used in the research community, (2) are implemented in Java, (3) contain relatively large numbers of classes, and (4) have publicly available source code. JHotDraw contains 508 concrete classes and is a Java graphics framework. Eclipse contains 1,571 concrete classes and is a multi-language software development environment that includes an integrated development environment and extensible plug-in system.

We selected the 17 cohesion measures summarized in Table A.1. These measures were considered by similar studies, which allows the current results to be compared with those obtained in previous studies.

We developed a Java tool to automate the cohesion measurement process for Java classes using the 17 cohesion measures considered here. The tool (1) parsed the source code of the Java classes, (2) extracted the data required to obtain cohesion values using the 17 measures, (3) recognized the constructors and access methods, and (4) calculated cohesion values for each of the four scenarios of considering or ignoring SPs. The tool obtained the corresponding cohesion values for each class in the two systems (a total of 2,079 classes). The empirical study in this paper did not explore the effects of considering destructors and delegation methods because Java does not support destructors and because delegation methods strongly rely on the problem semantics, which makes automatic identification of the delegation methods difficult. In contrast, both access methods and constructors can be automatically identified, and therefore, the empirical studies accounted for these methods. We considered only concrete classes (2,079 classes) and ignored abstract classes and interfaces because they did not have defined values for most considered measures.

For each considered scenario of considering or ignoring SPs, Table 1 reports descriptive statistics for the number of methods in the classes considered. Table 1 demonstrates that (1) when accounting for constructors, the average number of methods in a class (ANM) changed from 9.08 to 10.33 (13.4%); (2) when accounting for access methods, ANM was modified by 35.8%, from 9.08 to 12.33; and (3) when accounting for all SPs, ANM considerably increased by 46.1%, from 9.08 to 13.27. This significant change in the

ANM when considering SPs demonstrates the importance of exploring their effects on cohesion values. In addition, this significant change in the ANM potentially implies that a measure incorrectly quantifies the cohesion level when it accounts for all methods if the measure should only account for non-SPs.

Scenario ID	Scenario	Min	Max	25%	Med	75%	Mean	Std. Dev.
S_1	Ignoring all considered SPs	0	176	1	4	10	9.08	15.62
S_2	Considering constructors	0	177	3	5	11	10.33	15.77
S_3	Considering access methods	0	196	2	6	14	12.02	18.43
S_4	Considering all considered SPs	1	196	4	7	15	13.27	18.62

Table 1: Descriptive statistics regarding the number of methods in the classes considered.

3.2. Data Analysis Technique and Collection Process

We applied logistic regression (LR) [25], a statistical technique that is widely used to build prediction models for external quality attributes, such as class fault-proneness (e.g., [3, 16, 26, 27]), class reuse-proneness (e.g., [22]), and class maintainability (e.g., [28]). Applying other statistical techniques to construct prediction models, such as those used by Briand and Wuest [29], Subramanyam and Krishnan [30], and Arisholm et al. [31], and comparing the results with the LR results are outside the scope of this paper.

In LR analysis, the variables are classified as dependent or independent. The independent variables (i.e., the considered cohesion measures) are used to predict or explain the dependent variable (i.e., the actual instantiation reuse data). A dependent variable only takes a binary value. The value of the dependent variable depends on the prediction problem. For example, if it is required to predict the classes reused at least once, the binary variable will be set to "1" when the class is reused at least once and set to "0" otherwise. In the context of exploring the instantiation reuse-proneness prediction power of measures, the resulting models are intended to be applied to predict the classes that are reusable via instantiation. These classes must be well documented and tested; otherwise, their potential faults may negatively affect the other classes that instantiate these reusable classes. Typically, a large percentage of classes (e.g., 65% of the considered classes) are reused at least once via instantiation by other classes in the system. Consequently, it is impractical to devote more documentation and testing efforts to this relatively high percentage of classes. Alternatively, we decided to explore the abilities of the measures to predict IRI classes. We defined a class as intensively reused via instantiation when the number of times that the class is instantiated by the other classes in the system is greater than the average number of instantiations per class for all classes in the selected systems. We chose the average value because it is a representative statistical value. Accordingly, in our LR analysis, we set the dependent variable to "1" when the class is instantiated by the other classes in the considered systems a number of times that is greater than the average number of instantiations per class; otherwise, we set the value of the dependent variable to "0". Following Bieman and Kang [4] and Al Dallal and Morasca [22], we considered only instantiation reuse within the considered systems (i.e., private reuse) because it is extremely difficult to build models based on public reuse data [22].

We applied our own tool to analyze the two systems under consideration and to automate the detection of instantiations of each class. Among the considered classes, 11.6% of the classes had a number of instantiations greater than the average (i.e., the value of the dependent variable is set to "1" for these classes).

Once the LR analysis is applied, the following probability estimation equation is formed:

$$\pi(X_1, X_2, ..., X_n) = \frac{1}{1 + e^{-(C_0 + C_1X_1 + C_2X_2 + ... + C_nX_n)}}$$

In our context, π represents the probability of the class to be IRI, the X_is are the cohesion measures, and the C_i coefficients are estimated by maximizing the likelihood function extracted during the LR analysis [25]. C_0 is commonly called the *intercept*. The LR analysis can be performed using one or more independent variables. The former and latter types are referenced as univariate and multivariate LR, respectively. In this study, we did not intend to construct multivariate prediction models. Instead, for each considered cohesion measure, we intended to compare its abilities to predict instantiation reuse-proneness in the various scenarios considered. Achieving this goal requires performing univariate LR analysis only. When a single measure is considered in LR analysis, the probability estimation equation becomes

$$\pi(X) = \frac{1}{1 + e^{-(C_0 + C_1X)}}$$

In practice, the π of each class of interest must be calculated. The software engineer must pay more attention to classes with relatively high π values because these classes are candidates for being IRI. A threshold *t* must be set for π to classify the classes accordingly and to assess the classification performance of a probability estimation model. All classes whose estimated probabilities are greater than *t* are estimated to be IRI. Determining a threshold *t* for classification is a subjective choice that may dramatically change the classification results. In our analyses, we set *t* to be the proportion of actual positives in the considered dataset, i.e., a class is classified as being IRI if its estimated probability is greater than this proportion (0.116).

For each of the constructed univariate prediction models, we report the C_0 and C_1 coefficients and p-value (i.e., the probability that the C_1 value is different from zero by chance), where a p-value less than 0.05 indicates that the measure is a statistically significant predictor for class instantiation reuse-proneness. To evaluate the classification performance of the constructed models, we applied the receiver operating characteristic (ROC) curve [25] indicator, where ROC area is the area under the curve that represents the ratio of classes correctly classified as IRI versus the ratio of classes incorrectly classified as IRI at different thresholds. The ROC area is a better classification criterion than recall and precision because the ROC area assesses the classification performance regardless of the selected threshold *t*, whereas both recall and precision depend on the selected threshold *t*. According to the value of the ROC area, the classification performance is categorized as follows: ROC < 50% indicates that the classification is

bad, 50% ≤ ROC < 60% indicates that the classification is poor, 60% ≤ ROC < 70% indicates that the classification is fair, 70% ≤ ROC < 80% indicates that the classification is acceptable, 80% ≤ ROC < 90% indicates that the classification is excellent, and 90% ≤ ROC ≤ 100% indicates that the classification is outstanding. Practical models are those that have an ROC area of at least 70% [32].

To obtain more realistic and validated models, we applied the 10-cross-validation technique when constructing each univariate prediction model. For each model, we partitioned the considered data set into 10 subsamples and used each subsample to evaluate the classification performance of the regression model that was constructed using the remaining subsamples.

The probability estimation equation for a univariate model can be visualized using a curve that indicates the change in the π value as the cohesion value changes. For example, the hypothetical curves given in Figure 1 graphically represent the probability estimation equations of a measure in two scenarios. Based on a certain classification threshold *t*, the curves indicate that classes with cohesion values according to scenario *y* that are greater than a_1 are predicted to be IRI, whereas the classes of the other cohesion values are predicted to not be IRI. The chart in Figure 1 demonstrates that a class has different reuse-proneness prediction results for the two indicated scenarios in two cases. The first case is when the cohesion value of the class in scenario *x* is greater than a_2 and the cohesion value in scenario *y* is less than or equal to a_1. In this case, the class is predicted to be IRI in scenario *y* but not in scenario *x*. The other case is when the cohesion value of the class in scenario *x* is less than or equal to a_2 and the cohesion value in scenario *y* is greater than a_1. In this case, the class is predicted to be IRI in scenario *x* but not in scenario *y*. We refer to the measure value for which the prediction results change from one to zero, or vice versa, as the critical prediction value (CPV). For example, as depicted in Figure 1, the CPVs of the measure are a_1 and a_2 for scenarios *y* and *x*, respectively. Given a specific probability threshold (0.116 in our case), the *x* value (i.e., the CPV of a measure) can be calculated using the following equation, which is obtained from the probability estimation equation for a univariate model:

$$x = \frac{-\ln\left(\frac{1-t}{t}\right) - C_0}{C_1}$$

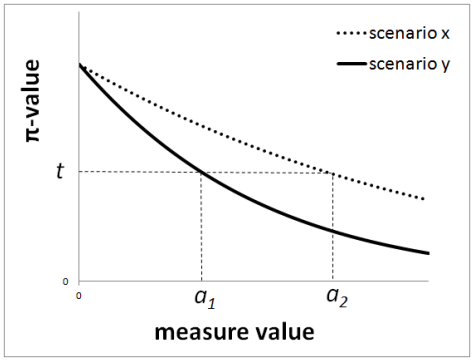

Figure 1: Hypothetical probability estimation curves for different scenarios.

We refer to a class for which its prediction results according to a certain cohesion measure differ from one scenario to another as a critically predicted class (CPC). The existence of CPCs provides evidence that the type of SPs considered can alter the prediction results.

Using Shapiro-Wilk normality test [33], we evaluated the distribution normality of the probability estimations obtained using each of the prediction models built for each considered measures in each scenario. None of them found to follow the normal distribution. Therefore, we applied the Wilcoxon paired test [34], which is a standard non-parametric statistical technique that is applied to compare two independent sets of samples that are not necessarily normal. Based on Wilcoxon test and according to the typical significance threshold ($\alpha = 0.05$), we considered the two sets to be insignificantly different (i.e., the change in the prediction results was statistically insignificant) when the corresponding α is greater than 0.05.

3.3. Effects of Accounting for SPs

We obtained univariate analysis results for each considered measure in each of the four scenarios of considering or ignoring SPs. Because of space limitations, Table 2 provides only the C_1 and ROC area results. To compare the classification performances of the individual measures, we focus on the ROC area results because, as mentioned earlier, the ROC area is a better evaluation criterion than standard precision and recall. For each measure reported in Table 2, we highlight the best ROC area result in boldface. In Table 2, the prediction results for certain scenarios are not included for those measures that were found to be insignificant predictors of instantiation reuse-proneness (i.e., they had a p-value < 0.05).

Measure	S_1: Considering SPs		S_2: Considering constructors		S_3: Considering access methods		S_4: Considering SPs	
	C_1	ROC area	C_1	ROC area	C_1	ROC area	C_1	ROC area
LCOM1	< 0.01	71.21	< 0.01	71.67	< 0.01	75.75	<0 .01	**76.38**
LCOM2	< 0.01	70.19	< 0.01	70.26	<0 .01	74.80	< 0.01	**75.28**
LCOM3	0.02	56.93	0.02	53.54	0.03	**65.38**	0.02	61.36
LCOM4	0.02	56.17	0.02	53.13	0.03	**64.78**	0.02	61.42
LCOM5	-	-	0.39	54.74	0.33	55.49	0.51	**59.61**
Coh	-1.92	68.38	-2.61	73.58	-2.55	73.67	-3.15	**76.79**
LSCC	-1.54	64.39	-1.80	66.43	-1.89	69.20	-2.20	**72.34**
CC	-1.51	63.14	-1.71	64.50	-1.85	66.46	-2.12	**69.11**
SCOM	-1. 60	64.38	-1.86	66.19	-1.97	69.11	-2.28	**71.81**
TCC	-0.40	49.06	-0.47	49.77	-0.87	53.38	-0.92	**53.81**
LCC	-	-	-	-	-	-	-	-
DC_D	-0.39	49.21	-0.38	49.20	-0.87	**53.77**	-0.82	53.29
DC_I	-	-	-	-	-	-	-	-
CBMC	-0.97	**47.97**	-1.92	46.70	-1.33	47.85	-2.34	46.51
ICBMC	-0.93	**47.96**	-1.90	46.70	-1.28	47.86	-2.36	46.51
OL_2	-0.97	**47.96**	-1.93	46.70	-1.34	47.86	-2.35	46.51
PCCC	-1.08	60.63	-1.57	63.10	-1.49	65.24	-2.19	**65.43**

Table 2: Univariate LR results for the SP scenarios

Figure 2 shows the probability estimation curves for eight of the considered measures (i.e., LCOM1, LCOM3, Coh, LSCC, TCC, LCC, CBMP, and PCCC) in the four scenarios of considering or ignoring SPs. The shapes of the curves for the other measures are not shown because of space limitations and because they are similar to those presented. Specifically, the shapes of the curves for LCOM2 are similar to those for LCOM1. The same behavior applies for the curves for LCOM4 compared with those for LCOM3, LCOM5 with Coh (which changed with the same rates but in different directions), CC and SCOM with LSCC, DC_D with TCC, DC_I with LCC, and ICBMC and OL_2 with CBMC. The considered classification threshold (0.116) is shown for each of the charts given in Figure 2. Finally, Table 3 reports the CPV for each measure in each scenario. In addition, for each considered pair of scenarios, Table 3 reports the percentage of CPCs and the p-values values obtained using the paired Wilcoxon technique. The p-values values for the insignificantly changed prediction decisions (i.e., p-value > 0.05) are highlighted in boldface in Table 3.

The following observations can be made based on the results reported in Tables 2 and 3 and Figure 2:

- **Significance of the prediction abilities**: Except for LCOM5 in Scenario 1 and both LCC and DC_I in all scenarios of considering or ignoring SPs, all measures were found to be significant predictors of IRI classes. Most of the significant predictor measures featured the best ROC area values in the fourth scenario. Most of the connectivity-based measures exhibited the best ROC area results for Scenario 1 (i.e., CBMC, ICBMC, and OL_2) and Scenario 3 (i.e., LCOM3 and LCOM4). Considering the best scenarios in terms of ROC area for the individual measures, LCOM1, LCOM2, Coh, LSCC, and SCOM were found to be acceptable predictors for IRI classes. The ROC area results for LCOM3, LCOM4, CC, and PCCC were within the fair range. The remainder of the significant predictor measures had either poor or bad classification performances. The charts shown in Figure 2 and the CPV values reported in Table 3 indicate that the measure values at which the prediction results change (i.e., the intersection points between the curves and the t π-value) are within the first quarter of the possible measure values for all connectivity-based measures (i.e., LCOM3, LCOM4, CBMC, ICBMC, and PCCC). For all other measures except LCOM1 and LCOM2, the CPV values are within the second quarter of possible values. LCOM1 and LCOM2 feature extreme outliers and thus do not have maximum values that can be used to decide in which quarter the corresponding CPVs are located. These observations indicated that classes must have relatively very low connectivity-based cohesion values to be estimated as IRI, which is not the case for cohesion measures that follow other measuring approaches.

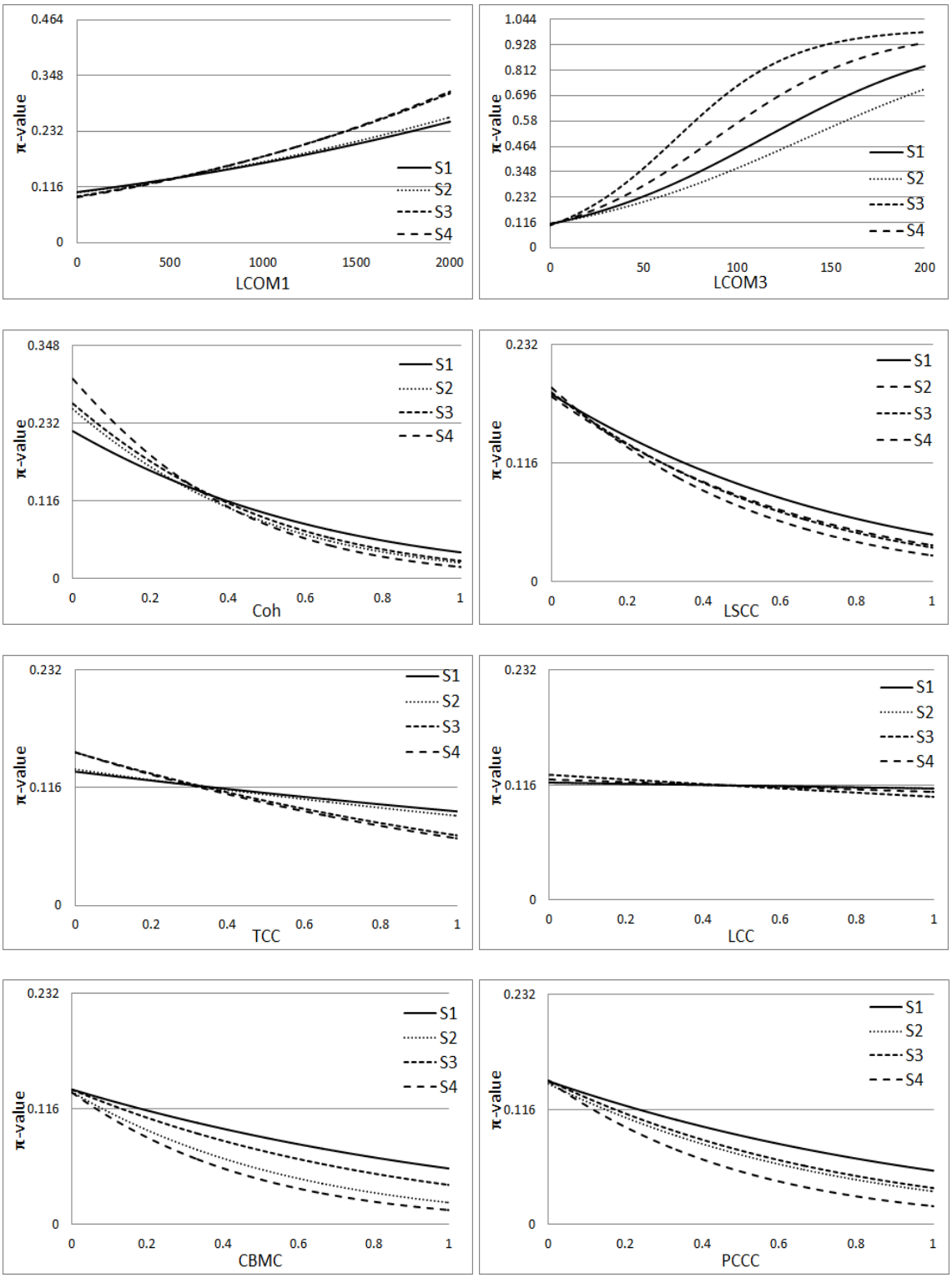

Figure 2: Probability estimation curves of different measures for different scenarios of considering or ignoring SPs.

Measure	CPV				S_1 vs. S_2		S_1 vs. S_3		S_1 vs. S_4	
	S_1	S_2	S_3	S_4	% of CPCs	p-value	% of CPCs	p-value	% of CPCs	p-value
LCOM1	209	220	291	310	0.82	0.000	2.65	< 0.0001	2.98	< 0.0001
LCOM2	198	207	269	284	0.29	< 0.0001	2.16	< 0.0001	2.31	< 0.0001
LCOM3	3.54	3.29	4.01	3.76	3.94	< 0.0001	9.81	< 0.0001	7.26	< 0.0001
LCOM4	3.42	3.27	3.92	3.74	3.90	0.001	8.47	< 0.0001	7.41	< 0.0001
LCOM5	1.00	0.96	0.88	0.87	13.42	0.022	22.17	< 0.0001	26.46	< 0.0001
Coh	0.40	0.36	0.39	0.37	7.89	< 0.0001	7.50	< 0.0001	10.58	<0.001
LSCC	0.35	0.29	0.29	0.26	7.84	< 0.0001	8.85	< 0.0001	11.54	< 0.0001
CC	0.38	0.32	0.34	0.31	7.65	< 0.0001	9.14	< 0.0001	11.16	< 0.0001
SCOM	0.37	0.31	0.31	0.29	8.71	< 0.0001	9.62	< 0.0001	11.74	< 0.0001
TCC	0.37	0.36	0.35	0.33	11.64	< 0.0001	9.04	**0.187**	13.66	**0.526**
LCC	0.41	0.40	0.45	0.44	12.65	**0.902**	8.90	< 0.0001	12.36	< 0.0001
DC_D	0.38	0.37	0.35	0.34	11.93	< 0.0001	9.09	**0.169**	13.08	**0.416**
DC_I	0.42	0.46	0.45	0.44	13.61	**0.809**	8.90	< 0.0001	12.79	< 0.0001
CBMC	0.18	0.08	0.13	0.06	11.40	< 0.0001	3.94	< 0.0001	12.17	< 0.0001
ICBMC	0.18	0.08	0.13	0.06	11.06	< 0.0001	4.67	< 0.0001	11.98	< 0.0001
OL_2	0.18	0.08	0.13	0.06	11.59	< 0.0001	4.04	< 0.0001	12.51	< 0.0001
PCCC	0.24	0.15	0.17	0.12	9.52	< 0.0001	9.72	< 0.0001	14.62	< 0.0001

Table 3: CPVs, percentage of CPCs, and Wilcoxon p-values values for the considered pairs of scenarios of considering or ignoring SPs

- **Direction of impact on instantiation reuse-proneness**: In all scenarios, the signs of the C_1 values, which are reported in Table 2, and the trends of change, which are shown in Figure 2, for all considered measures demonstrated that the cohesion measures negatively impacted instantiation reuse-proneness and that the lack of cohesion measures positively impacted instantiation reuse-proneness. The results confirm the expectation that the impact of the lack of cohesion measures, which measure the absence of cohesion, on external attributes is opposite that of cohesion measures, which measure the presence of cohesion. In all cases, the results indicated that the cohesion internal attribute negatively impacted the instantiation reuse-proneness external attribute. The consideration of any type of SPs did not change the direction of impact of any of the significant predictor measures on instantiation reuse-proneness.
- **Significance of the changes**: Table 3 indicates that most prediction results significantly changed (i.e., Wilcoxon p-value < 0.05) when any type of SPs was considered compared to the scenario in which all SPs were ignored. Therefore, we rejected null hypotheses 6, 7, and 8 and concluded that, with few exceptions, selecting the scenario empirically determined to be the best for considering or ignoring SPs is expected to significantly improve the prediction results. The percentage of CPCs ranged from 0.82% to 13.61%, with an average of 8.7%, when considering constructors; from 2.16% to 22.17%, with an average of 8.15%, when considering access methods; and from 2.31% to 26.46%, with an average of 11.45%, when considering both access methods and constructors. These significant average percentages of CPCs indicate that considering SPs potentially affects the abilities of measures to predict IRI classes.

3.4. Discussion

Taking into account all scenarios for considering or ignoring SPs, most of results indicate that cohesion is a significant predictor of the instantiation reuse-proneness external attribute. The results all confirm that cohesion negatively impacts the abilities of the cohesion measures to predict IRI classes. One of the possible interpretations for these observations is that less cohesive classes may be more prone to include extensive sets of functionalities. Such classes may be considered more useful and ready to instantiate. These results appear to be inconsistent with the corresponding results obtained by Al Dallal and Morasca [22]. However, it is important to note that this paper considers a problem different from that considered by Al Dallal and Morasca [22]. Here, we consider the prediction of *intensively* reused classes, whereas the other work considered the prediction of classes that are reused at least once by instantiation.

When using a cohesion measure in a prediction model for an external quality attribute, software practitioners are not interested in applying the version of the cohesion measure that provides the best cohesion indication. Instead, they are interested in applying the version of the measure that provides the best prediction results for the external quality attribute. The obtained results demonstrate that considering or ignoring SPs significantly affected the measures' abilities to predict IRI classes. Table 4 summarizes the related null hypotheses 1-3 and the corresponding study results and conclusions. These results indicate that there is a relationship between SPs and instantiation reuse-proneness. We believe that one of these relationships is that when a class is instantiated, at least one of the constructors must be invoked. A class with multiple constructors provides more flexibility for initializing the attributes of the class, which makes the class more appealing for reuse. In addition, the existence of access methods in the class is expected to make the class more appealing for reuse via instantiation because access methods provide the flexibility to indirectly use attributes that are declared private in the instantiated class. These expectations imply that when using a cohesion measure, both types of SPs must be considered to better predict instantiation reuse-proneness. Otherwise, the cohesion measure may mistakenly miss a factor that has a positive or negative impact on instantiation reuse-proneness.

4. Threats to Validity

The external and internal threats to validity of the reported empirical studies are detailed and discussed as follows.

4.1. Internal Validity

Quality attributes other than cohesion must be considered to predict reusable classes via instantiation [22]. However, constructing optimal prediction models for IRI classes is not one of our goals here. Instead, our studies aimed at exploring whether there is empirical evidence that considering or ignoring SPs in cohesion measurement can inadequately impact instantiation reuse-proneness of classes, which is an application of interest for software developers and practitioners.

Id	Null-hypothesis	Null-hypothesis study result	Conclusion
1	Considering constructors in cohesion measurement does not significantly change the abilities of cohesion measures to predict IRI classes.	Rejected for most of the considered measures	Considering constructors in cohesion measurement could significantly change the abilities of most of the considered cohesion measures to predict IRI classes.
2	Considering access methods in cohesion measurement does not significantly change the abilities of cohesion measures to predict IRI classes.	Rejected for most of the considered measures	Considering access methods in cohesion measurement could significantly change the abilities of most of the considered cohesion measures to predict IRI classes.
3	Considering SPs in cohesion measurement does not significantly change the abilities of cohesion measures to predict IRI classes.	Rejected for most of the considered measures	Considering SPs in cohesion measurement could significantly change the abilities of most of the considered cohesion measures to predict IRI classes.

Table 4: Summary of the null-hypotheses regarding the effect of accounting for SPs on the abilities of the considered measures to predict IRI classes.

In the instantiation reuse-proneness prediction empirical study, we chose a threshold to demonstrate the effects of considering SPs in cohesion measurement on the prediction ability. The threshold was based on the percentage of the actual number of classes reused via instantiation. Selecting another threshold may change the percentage of classes that their prediction results change when SPs are considered. However, the empirical study showed that, at least for this representative threshold, the consideration of SPs in cohesion measurement caused the prediction results to significantly change.

When constructing the LR models, we considered a class to be IRI when it is reused a number of times greater than the average number of times a considered class, in the considered systems, is instantiated. Although other threshold values could be considered, the selected value is a typical statistical representative value and it led to building prediction models with significant abilities to predict IRI class. This empirical study considered only local reuse-proneness. Considering the reuse-proneness of a class outside the system under consideration might be uncontrollable and requires an exhaustive search on the Internet and in other resources [22].

4.2. External Validity

We only accounted for Java systems, which disallowed us from empirically exploring the impact of considering destructors. However, constructors and destructors are expected to have similar effects [8, 35]. In this paper, we considered two well-known and widely used systems within the research community; Eclipse and JHotDraw. Although these systems are open-source systems that may not be representative of all industrial domains, conducting empirical studies using open-source systems is a common practice within the research community. To obtain more generalized results, it is required to perform similar large-scale empirical studies that engage large-scale systems written in different programming languages and chosen from different domains.

5. Conclusions and Future Work

This paper empirically considered the effects of considering or ignoring SPs in cohesion measurement on the abilities of the cohesion measures to predict IRI classes. Two types of SPs were considered: constructors and access methods. Four different scenarios to cover the different possibilities of considering or ignoring SPs were considered for each of the 17 cohesion measures selected. The empirical studies considered classes of two relatively large and well-known Java systems.

We performed an empirical study to explore the impact of considering SPs in cohesion measurement when predicting IRI classes. The results demonstrated that considering SPs in cohesion measurement caused the prediction results to change significantly. The best IRI class predictor for most of the considered measures was the model for the scenario that included all SPs. This scenario is thus recommended when incorporating cohesion measures into prediction models for classes that are likely to be reused via instantiation.

The results obtained and conclusions drawn from this empirical study have both similarities and dissimilarities with the corresponding results and conclusions in previous empirical studies that used different datasets. The general conclusions about the relationships between the cohesion internal quality attribute and instantiation reuse-proneness external quality attribute were different than those obtained by Bieman and Kang [4] and Al Dallal and Morasca [22]. Here, we found that most of the cohesion measures were significant predictors of classes reused via instantiation and that the relation between cohesion and instantiation reuse-proneness was negative. Conversely, Al Dallal and Morasca [22] found that most of the cohesion measures were insignificant predictors of class reuse via instantiation and that the relation between cohesion and instantiation reuse-proneness is positive. The reason behind these differences is that we modified the considered problem such that the cohesion measures became more useful for reuse-proneness prediction. Our modifications included changing the problem to consider the prediction of intensively reused classes instead of predicting classes reused at least once via instantiation. The problem addressed here is more practical than the other problem because considering this problem enables software engineers to spend more documentation and testing time and effort on a reduced set of classes.

In this paper, we considered an application of interest that differs from those considered by Al Dallal [8]. It was thus expected that we would come to different conclusions about whether to consider or ignore SPs in cohesion measurement when using a cohesion measure as part of a prediction model for an external quality attribute. Unlike this study, the results of empirical study performed by Al Dallal [8] indicated that the fault-proneness prediction abilities of the cohesion measures did not significantly change when SPs were considered; the recommendation was to include constructors and exclude access methods.

This paper focused on exploring the effect of considering or ignoring SPs on the abilities of the cohesion measures to predict class instantiation reuse-proneness. Exploring this

effect on the prediction results for other types of class reuse, such as reuse via inheritance, or for other external quality attributes, such as testability and maintainability, is left to future research. Finally, the empirical study reported in this paper can be replicated using different sets of cohesion measures and classes from other systems to confirm or deny the obtained results and conclusions.

Acknowledgments
The author would like to acknowledge the support of this work by Kuwait University Research Grant WI01/12. In addition, the author would like to thank Anas Abdin for assisting in collecting the cohesion results.

Appendix A: Definitions of the considered cohesion measures

Class Cohesion Measure	Definition/Formula
The lack of Cohesion in Methods (LCOM1) [36]	LCOM1= Number of pairs of methods that do not share attributes.
LCOM2 [37]	*P*= Number of pairs of methods that do not share attributes. *Q*= Number of pairs of methods that share attributes. $LCOM2 = \begin{cases} P-Q & if\ P-Q \geq 0 \\ 0 & otherwise \end{cases}$
LCOM3 [38]	LCOM3= Number of connected components in the graph that represents each method as a node and the sharing of at least one attribute as an edge.
LCOM4 [39]	Similar to LCOM3, where additional edges are used to represent method invocations.
LCOM5 [40]	LCOM5= (*kl-a*)/(*kl-l*), where *l* is the number of attributes, *k* is the number of methods, and *a* is the sum of the numbers of distinct attributes that are accessed by each method in a class.
Coh [3]	Coh=*a/kl*, where *a*, *k*, and *l* have the same definitions as above.
Low-level design Similarity-based Class Cohesion (LSCC) [41]	$LSCC(C) = \begin{cases} 0 & \text{if } l = 0 \text{ and } k > 1, \\ 1 & \text{if } (l > 0 \text{ and } k = 0) \text{ or } k = 1, \\ \dfrac{\sum_{i=1}^{l} x_i (x_i - 1)}{lk(k-1)} & \text{otherwise.} \end{cases}$, where *l* is the number of attributes, *k* is the number of methods, and x_i is the number of methods that reference attribute *i*.
Class Cohesion (CC) [42]	CC= The ratio of the sum of the similarities between all pairs of methods to the total number of pairs of methods. The similarity between methods *i* and *j* is defined as $Similarity(i, j) = \dfrac{\lvert I_i \cap I_j \rvert}{\lvert I_i \cup I_j \rvert}$, where I_i and I_j are the sets of attributes that are referenced by methods *i* and *j*, respectively.
Class Cohesion Measure (SCOM) [18]	SCOM= The ratio of the sum of the similarities between all pairs of methods to the total number of pairs of methods. The similarity between methods *i* and *j* is defined as $Similarity(i, j) = \dfrac{\lvert I_i \cap I_j \rvert}{\min(\lvert I_i \rvert, \lvert I_j \rvert)} \cdot \dfrac{\lvert I_i \cup I_j \rvert}{l}$, where *l* is the number of attributes.
Tight Class Cohesion (TCC) [4]	TCC= The relative number of directly connected pairs of methods, where two methods are directly connected if they are directly connected to an attribute. A method *m* is directly connected to an attribute when the attribute appears within the body of the method or within the body of a method that is directly or transitively invoked by method *m*.
Loose Class Cohesion (LCC) [4]	LCC= The relative number of directly or transitively connected pairs of methods, where two methods are transitively connected if they are directly or indirectly connected to an attribute. A method *m*, which is directly connected to an attribute *j*, is indirectly connected to an attribute *i* when there is a method that is directly or transitively connected to both attributes *i* and *j*.

Table A.1: Definitions of the class cohesion measures under consideration

Class Cohesion Measure	Definition/Formula
Degree of Cohesion-Direct (DC_D) [17]	DC_D= The relative number of directly connected pairs of methods, where two methods are directly connected if they satisfy the condition mentioned above for TCC or if the two methods directly or transitively invoke the same method.
Degree of Cohesion-Indirect (DC_I) [17]	DC_I= The relative number of directly or transitively connected pairs of methods, where two methods are transitively connected if they satisfy the same condition mentioned above for LCC, or if the two methods directly or transitively invoke the same method.
Cohesion Based on Member Connectivity (CBMC) [5]	CBMC(G)=F_c(G)×F_s(G), where F_c(G)=|M(G)|/|N(G)|, M(G)=number of glue methods in graph G, N(G)=number of nonspecial methods in graph G, $F_s(G)=[\sum_{i=1}^{n} CBMC(G^i)]/n$, n=number of child nodes of G, and the set of glue methods is the minimal set of methods such that its removal causes the class representative graph to become disconnected.
Improved Cohesion Based on Member Connectivity (ICBMC) [43, 44]	ICBMC(G)=F_c(G)×F_s(G), where F_c(G)=|M(G)|/|N(G)|, M(G)=the number of edges in the cut set of G, N(G)=the number of non-special methods represented in graph G multiplied by the number of attributes, $F_s(G)=[\sum_{i=1}^{2} ICBMC(G^i)]/2$, and cut set is the minimum set of edges such that their removal causes the graph to become disjointed..
OL_n [45]	OL_n= the average strength of the attributes, where the strength of an attribute is the average strength of the methods that reference that attribute. The strength of a method is initially set to 1 and is computed in each iteration as the average strength of the attributes that it references, where *n* is the number of iterations used to compute OL.
Path Connectivity Class Cohesion (PCCC) [46]	$PCCC(C)=\begin{cases}0 & \text{if } l=0 \text{ and } k>1,\\ 1 & \text{if } l>0 \text{ and } k=0,\\ \frac{NSP(G_c)}{NSP(FG_c)} & \text{otherwise.}\end{cases}$, where NSP is the number of simple paths in graph G_c, FG_c is the corresponding fully connected graph, and a simple path is a path in which each node occurs once at most.

Table A.1 (continuation): Definitions of the class cohesion measures under consideration

References

[1] S. Morasca, A probability-based approach for measuring external attributes of software artifacts, *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, USA, pp. 44-55.

[2] N. Fenton and S. Pfleeger, Software Measures: A Rigorous and Practical Approach, Course Technology, 2nd edition, 1998.

[3] L. C. Briand, J. Daly, and J. Wuest, A unified framework for cohesion measurement in object-oriented systems, *Empirical Software Engineering - An International Journal*, 3(1), 1998, pp. 65-117.

[4] J. Bieman and B. Kang, Cohesion and reuse in an object-oriented system, *ACM SIGSOFT Software Engineering Notes*, 20(SI), 1995, pp. 259-262.

[5] H. S. Chae, Y. R. Kwon, and D. Bae, A cohesion measure for object-oriented classes, *Software—Practice & Experience*, 30(12), 2000, pp.1405-1431.

[6] L. Etzkorn, C. Davis, and W. Li, A practical look at the Lack of Cohesion in Methods measure, *Journal of Object-Oriented Programming*, 11(5), 1998, pp. 27-34.

[7] Y. Zhou, J. Lu, H. Lu, and B. Xu, A comparative study of graph theory-based class cohesion measures, ACM SIGSOFT Software Engineering Notes, 29(2), 2004, pp. 13-13.

[8] J. Al Dallal, The impact of accounting for special methods in the measurement of object-oriented class cohesion on refactoring and fault prediction activities, *Journal of Systems and Software*, 2012, 85(5), pp. 1042-1057.

[9] J. Al Dallal, The Incorporating transitive relations in low-level design-based class cohesion measurement, *Software—Practice & Experience*, 2013, Vol. 43. No. 6, pp. 685-704.
[10] J. Al Dallal, Mathematical validation of object-oriented class cohesion measures, *International Journal of Computers*, 4(2), 2010, pp. 45-52.
[11] J. Al Dallal, Improving the applicability of object-oriented class cohesion measures, *Information and Software Technology*, 2011, 53(9), pp. 914-928.
[12] J. Al Dallal, Measuring the discriminative power of object-oriented class cohesion measures, *IEEE Transactions on Software Engineering*, 2011, 37(6), pp. 788-804.
[13] J. Al Dallal, Constructing models for predicting extract subclass refactoring opportunities using object-oriented quality measures, *Information and Software Technology*, 2012, 54(10), pp 1125-1141.
[14] J. Al Dallal and L. Briand, An object-oriented high-level design-based class cohesion measure, *Information and Software Technology*, 2010, 52(12), pp. 1346-1361.
[15] L. C. Briand, J. Wust, J. Daly, and V. Porter, Exploring the relationship between design measures and software quality in object-oriented systems, *Journal of System and Software*, 51(3), 2000, pp. 245-273.
[16] L. C. Briand, J. Wüst, and H. Lounis, Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs, *Empirical Software Engineering*, 6(1), 2001, pp. 11-58.
[17] L. Badri and M. Badri, A Proposal of a new class cohesion criterion: an empirical study, *Journal of Object Technology*, 3(4), 2004, pp. 145-159.
[18] L. Fernández, and R. Peña, A sensitive measure of class cohesion, *International Journal of Information Theories and Applications*, 13(1), 2006, pp. 82-91.
[19] J. Bieman, Deriving measures of software reuse in object oriented systems, *Proceedings of the BCS-FACS Workshop on Formal Aspects of Measurement*, London, 1992, pp.63-83.
[20] J. Barnard, A new reusability measure for object-oriented software, *Software Quality Journal*, 7(1), 1998, pp.35-50.
[21] Y. Lee and K. Chang, Reusability and maintainability measures for object-oriented software, *Proceedings of the 38th annual on Southeast regional conference*, USA, 2000.
[22] J. Al Dallal and S. Morasca, Predicting object-oriented class reuse-proneness using internal quality attributes, *Empirical Software Engineering*, 19(4), 2014, pp. 775-821.
[23] JHotDraw, http://sourceforge.net/projects/jhotdraw/, accessed November 2013.
[24] Eclipse, http://www.eclipse.org/, accessed in November 2013.
[25] D. Hosmer and S. Lemeshow, *Applied Logistic Regression*, Wiley Interscience, 2000, 2nd edition.
[26] T. Gyimothy, R. Ferenc, and I. Siket, Empirical validation of object-oriented measures on open source software for fault prediction, *IEEE Transactions on Software Engineering*, 3(10), 2005, pp. 897-910.
[27] A. Marcus, D. Poshyvanyk, and R. Ferenc, Using the conceptual cohesion of classes for fault prediction in object-oriented systems, *IEEE Transactions on Software Engineering*, 34(2), 2008, pp. 287-300.

[28] J. Al Dallal, Predicting object-oriented class maintainability using internal quality attributes, *Information and Software Technology*, 2013, 55(11), pp. 2028-2048.
[29] L. C. Briand and J. Wust, Empirical studies of quality models in object-oriented systems, *Advances in Computers*, Academic Press, 2002, pp. 97-166.
[30] R. Subramanyam and M. Krishnan, Empirical analysis of CK measures for object-oriented design complexity: implications for software defects, *IEEE Transactions on Software Engineering*, 29(4), 2003, pp. 297-310.
[31] E. Arisholm, L. Briand, and E. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models, *Journal of Systems and Software*, 2010, 83(1), pp. 2-17.
[32] R. Shatnawi, W. Li, J. Swain, and T. Newman, Finding software metrics threshold values using ROC curves, *Journal of Software Maintenance and Evolution: Research and Practice*, 2010, 22(1), pp. 1-16.
[33] W. Rosenkrantz, Introduction to Probability and Statistics for Science, Engineering, and Finance, Chapman and Hall/CRC, 1 edition, 2008.
[34] J. Devore, Probability and Statistics for Engineering and the Sciences, Cengage Learning, 8 edition, 2011.
[35] J. Al Dallal, Qualitative analysis for the impact of accounting for SPs in object-oriented class cohesion measurement, *Journal of Software*, 2013, 8(2), pp. 327-336.
[36] S.R. Chidamber and C.F. Kemerer, Towards a Measures Suite for Object-Oriented Design, *Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, Special Issue of SIGPLAN Notices, 26(10), 1991, pp. 197-211.
[37] S.R. Chidamber and C.F. Kemerer, A Measures suite for object Oriented Design, *IEEE Transactions on Software Engineering*, 20(6), 1994, pp. 476-493.
[38] W. Li and S.M. Henry, Maintenance measures for the object oriented paradigm. *In Proceedings of 1st International Software Measures Symposium*, Baltimore, MD, 1993, pp. 52-60.
[39] M. Hitz and B. Montazeri, Measuring coupling and cohesion in object oriented systems, *Proceedings of the International Symposium on Applied Corporate Computing*, 1995, pp. 25-27.
[40] B. Henderson-sellers, *Object-Oriented Measures Measures of Complexity*, Prentice-Hall, 1996.
[41] J. Al Dallal and L. Briand, A Precise method-method interaction-based cohesion measure for object-oriented classes, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2012, 21(2), pp. 8:1-8:34.
[42] C. Bonja and E. Kidanmariam, Measures for class cohesion and similarity between methods, *Proceedings of the 44th Annual ACM Southeast Regional Conference*, Melbourne, Florida, 2006, pp. 91-95.
[43] B. Xu and Y. Zhou, Comments on 'A cohesion measure for object-oriented classes' by H. S. Chae, Y. R. Kwon and D. H. Bae (Softw. Pract. Exper. 2000, 30: 1405-1431), *Software—Practice & Experience*, Vol. 31, No. 14, 2001, pp. 1381-1388.
[44] B. Xu and Y. Zhou, More comments on 'A cohesion measure for object-oriented classes' by H. S. Chae, Y. R. Kwon and D. H. Bae (Softw. Pract. Exper. 2000, 30: 1405-1431), *Software—Practice & Experience*, Vol. 33, No. 6, 2003, pp.583-588.

[45] X. Yang, *Research on Class Cohesion Measures*, M.S. Thesis, Department of Computer Science and Engineering, Southeast University, 2002.
[46] J. Al Dallal, Fault prediction and the discriminative powers of connectivity-based object-oriented class cohesion measures, *Information and Software Technology*, 2012, 54(4), pp. 396-416.

Biography

Jehad Al Dallal received his PhD in Computer Science from the University of Alberta in Canada and was granted the award for best PhD researcher. He is currently working at Kuwait University in the Department of Information Science as an Associate Professor and he is currently the chairman of the department. Dr. Al Dallal has completed several research projects in the areas of software testing, software measures, and communication protocols. In addition, he has published more than 70 papers in reputable journals and conferences. Dr. Al Dallal was involved in developing more than 20 software systems. He also served as a technical committee member of several international conferences and as an associate editor for several refereed journals.