J Braz Comput Soc (2012) 18:251–266
DOI 10.1007/s13173-011-0052-4

ORIGINAL PAPER

Package level cohesion measurement in object-oriented software

Varun Gupta · Jitender Kumar Chhabra

Received: 25 December 2010 / Accepted: 6 December 2011 / Published online: 22 December 2011

Abstract Packages are re-usable components for most of object-oriented systems. To promote reuse in object-oriented systems and to make deployment and maintenance tasks easy, packages in object-oriented systems must be cohesive. Quantification of cohesion of packages can be very useful in assessing their reusability, quality etc. In this paper, a new measure for the measurement of package cohesion is proposed. The cohesion of a package is measured in terms of the degree of intra-package dependencies among its elements. The hierarchical structure of packages has also been taken into account during the measurement. The proposed measure has been validated theoretically as well as empirically. An empirical study has been conducted using 25 packages taken from six open-source software projects developed in Java. The proposed package cohesion measure is found to be a useful indicator of external quality factors such as the reusability of packages. The proposed metric is also established as a better predictor of code reusability than the existing cohesion measures.

Keywords Cohesion · Metrics · Quality · Reusability · Packages · Object-oriented software

1 Introduction

Major cost of a software system is devoted to its maintenance [49]. During maintenance, software professionals spend at least half their time reading and analyzing software in order to understand it [8, 28]. Classes contain abstractions of code elements that are essential for understanding and maintaining the system. However, they are too small to gain understanding of a system. Maintenance can be facilitated by organizing classes into highly modular groups, i.e., where the group contains only related classes and coupling between groups is minimized. In such groups it is easier to retrieve related items, and they facilitate understanding of the system. Such groups are termed as packages in modern object-oriented systems [60]. Packages consist of elements such as classes or interfaces which are conceptually interrelated to each other. The relations among elements of a package determine the cohesion of a package [58]. Cohesive packages provide a modular structure in which fine-grained classes can be assembled to provide coarse-grained functionality and thus, ease the maintenance and promote the reusability [59]. Packages are important because they are units of organization for object-oriented software systems [51] and are widely used in object-oriented languages such as Java [37], C# [64].

Packages should follow the basic principle of design-maximum cohesion and minimum coupling. This paper primarily targets measuring cohesion of a package, so as to maximize it later, resulting in a better quality component. “You cannot control what you cannot measure” [31], stresses the need of such measures. Thus a good maintainable and quality software system can be facilitated either by reusing the existing packages or by organizing the newly developed classes into packages, where the packages contain only interrelated classes and the cohesion of these packages is maximized. Moreover, packages lacking cohesion contain classes that independently perform numerous disjointed functions. If such non-cohesive packages are made available to the designers and programmers of the software, they will

V. Gupta (✉) · J.K. Chhabra
Department of Computer Engineering, National Institute of Technology, Kurukshetra 136119, India
e-mail: varun3dec@yahoo.com

J.K. Chhabra
e-mail: jitenderchhabra@rediffmail.com

have to import classes from many packages to provide full functionality, resulting into more efforts, and difficult maintenance. Another repercussion of this will be a more complex deployment or re-deployment of application services because multiple packages must be re-compiled, tested, and distributed. Thus, a good system design should consist of packages with high cohesion and low coupling among packages [63]. Still, only a few quantitative studies of the concrete use of cohesion and coupling have been conducted at the package level. In this paper, a new metric is proposed for measuring cohesion at the package-level in order to achieve good quality packages.

This paper first provides some basic definitions and properties regarding object-oriented systems containing packages and then defines cohesion measure at package level for object-oriented systems. While defining the metric for cohesion at package level, hierarchical structure of the packages has also been taken into consideration. The proposed measure is validated theoretically using Briand et al.'s evaluation criteria [13] and usefulness of the proposed metric is also established by correlating this metric with the external quality factor-reusability. This paper is organized as follows. In Sect. 2, the previous related works are reviewed and Sect. 3 provides the theoretical framework for package cohesion measurement. Section 4 contains the definition of the proposed measure and Sect. 5 presents the theoretical validation of the proposed measure. Section 6 validates the proposed measure through experimental evaluation conducted using open-source software projects and Sect. 7 presents the conclusions and future work directions.

2 Related work

In literature, cohesion metrics have been defined for modules in structured programming and for classes in object-oriented systems [10–12, 14, 23–26, 30, 32, 34, 40, 56, 57, 72, 73]. The metrics exist in literature for the measurement of cohesion at the higher levels of abstraction also [1, 3, 35, 38, 47, 50, 51, 54, 58, 63, 70, 74]. The approaches taken to measure cohesiveness of procedural programs have generally tried to evaluate cohesion on a procedure (function). Emerson proposed a measure to compute cohesion applicable to modules in the sense of Pascal procedures [34]. This measure was based on a graph theoretic property that quantified the relationship between control flow paths and references to variables. Bieman and Kang provided intra-module cohesion measures for cohesion based on the design-level information [11]. Another approach for module cohesion measurement using the slice abstraction of a program based on data slices was proposed by Bieman and Ott [12].

Most existing approaches for class cohesion measurement consider interactions between methods and/or attributes in a class. Chidamber and Kemerer defined cohesion of a class as the degree of similarity of its methods [25, 26]. They presented a measure for class cohesion named Lack of Cohesion in Methods (LCOM), improved by Henderson-Sellers's LCOM* [39]. Eder et al. [32] adapted the existing frameworks for cohesion in the procedural and object-based paradigm to the specifics of the object-oriented paradigm. They distinguished between three types of cohesion in an object-oriented system: method, class and inheritance cohesion and various degrees of cohesion were defined for each type of cohesion. Ott et al. adapted the approach of module cohesion measurement based on data slices for the cohesion measurement of a class in object-oriented systems [56, 57]. Bieman and Kang measured cohesion for classes using the number of pairs of methods in a class that access common instance variables [10]. They defined Tight Class Cohesion (TCC) and Loose Class Cohesion (LCC) measures based on the concept of direct and indirect common attribute usage by the public methods of the class. Bansiya and Davis came up with another metric, Cohesion Among Methods of Classes (CAMC) which evaluated the relatedness of methods in the interface of a class using the parameter lists defined for the methods [4, 5]. A significant advantage of the CAMC metric over other traditional metrics was that the new metric was not dependent on the implementation of the methods of a class and thus could be used during the design phase to measure the cohesiveness of methods in a class. Using a different approach, Counsell et al. proposed a metric based on the Hamming distance to demonstrate problems in cohesion measurement and concluded that cohesion and coupling measurement are interrelated [29]. Briand et al. defined a cohesive interaction graph to represent the design-level interactions and proposed a cohesion metric suite based on the graph [16, 73].

Metrics have been proposed for the measurement of cohesion at the higher levels of abstraction such as package, sub-systems and systems levels. Music measured the cohesion of a package as an external property of a module [52] and claimed that the internal organization of a module is not enough to determine its cohesion. Morris followed this line by computing module cohesion considering the fan-in of the contained objects [54]. Similarly, Anquetil and Lethbridge proposed Modularization Quality (MQ) [2] which used the dependencies between modules of two distinct subsystems and the ones between the modules of the same subsystem to determine the cohesion of clusters. Also, Brito e Abreu and Goulão [19] used cluster analysis techniques to obtain better modularization solutions as far as coupling and cohesion are concerned.

Patel et al. [58] employed the concept of document similarity to measure the similarity between subprograms by using the shared data types and the cohesion is computed to be the average of the similarity measures over distinct pairs of subprograms of Ada programs. Xu et al. [70] also proposed cohesion measures for packages in language Ada95. Martin proposed a package level metric set including Relational Cohesion (RC) which is defined as the ratio of the number of data relations in a package to the number of components in the package [51]. Using the package level metrics given by Martin, Atole and Kale assessed the quality of object-oriented design [3]. As opposed to counting, Allen et al. proposed an information-theory-based metric to evaluate module cohesion [1]. They first used a graph to represent the relations between elements in a module. Then, they regarded such a graph as an information source and modeled the pattern of edges incident to each node as a random variable. Finally, module cohesion was defined as excess entropy (EEC) [1]. It is easy for EEC to be applied to package, since it only requires intramodule-edges graph. However, both RC and EEC only consider intra-package data dependences, thereby not being adapted to the packages not having such kind of dependence. Lee and Liang proposed a cohesion measure, Information-based cohesion, (ICH) for a set of classes based on information flow through method invocations within classes [47]. They defined cohesion of a set of classes as the sum of the cohesion of the classes in the set. Using a similar approach, Gui and Scott [38] proposed system level cohesion measures for evaluation of component reusability and defined cohesion of the system as mean cohesion of all the classes of the system. Further, Tagoug [63] proposed cohesion measures for subjects (which are quite similar to the packages) and defined subject cohesion on the basis of interactions among classes within a subject. However, the author did not consider the hierarchy of subjects while defining the cohesion measure. Besides this, Ponisio et al. proposed an approach to measure package cohesion based on client usage rather than explicit dependencies within a package [59, 60]. Following a similar approach, Zhou et al. proposed a new measure called SCC for measuring semantic cohesion of a package by considering the fact that two components of a package are related tightly if they have similar contexts [73].

Software cohesion metrics proposed for object-oriented software do not cover the abstractions and complexity dimensions introduced by the aspect paradigm. As a consequence, some measures [20, 21, 36, 46, 61, 71] have been proposed for cohesion measurement for aspect-oriented software systems. Zhao and Xu approach [71] is the first proposal in the field of aspect cohesion measurement. It is based on a dependency model for aspect-oriented software that consists of a group of dependency graphs. According to the Zhao and Xu approach, cohesion is defined as the degree of relatedness between attributes and modules. Zhao and Xu presented an approach for measuring aspect cohesion based on inter-attributes, inter-modules and module-attribute dependencies. However, Sant'Anna et al. proposed in [61] an extension of the well-known LCOM (Lack of Cohesion in Methods) metric [26]. The proposed metric LCOO (Lack of Cohesion in Operations) measures the amount of method/advice pairs that do not access to the same instance variables. This metric measures the lack of cohesion of a component. Afterwards, Gélinas et al. proposed an approach for aspect cohesion measurement based on dependencies analysis [36]. They introduced several cohesion criteria taking into account aspects' features and capturing various dependencies between their members. They also proposed new aspect cohesion metric and compared it with other existing aspect cohesion metrics.

The proposed package level cohesion metrics in this paper define cohesion of a package in terms of the degree of intra-package dependencies among its classes not as the sum of the cohesion of the classes in the package as done by the existing measures such as ICH metric proposed by Lee and Liang [47] and metrics given by Gui and Scott [38]. Moreover, the hierarchical structure of a package has also been taken into account during cohesion measurement of the package which has been ignored by the existing cohesion measures such as metrics proposed by Tagoug [63]. Further, in Sect. 6.6, the proposed metric has been compared experimentally with the existing cohesion metrics LCOM [25, 26], LCOM* [39], TCC [10], CAMC [4, 5], RC [51], EEC [1], ICH [47], and SCC [74] using Karl Pearson correlation method [27, 45] to compute the correlation coefficients between the package reusability ratings given by a team of developers (as given in Table 4) and the values produced by the various cohesion measures. From the results of the study, it is found that the proposed measure produced the highest magnitude of correlation with the package reusability in comparison to the various cohesion measures. These results suggest that the proposed package level cohesion measure is a better indicator of reusability than the existing cohesion measures.

3 Theoretical framework for measurement

In this section, we provide basic definitions and properties related to packages, which are required to lay foundation of a rigorous framework for measurement of package cohesion. First, definitions related to the concept of packages are given and then, relationships between elements of a package are defined.

3.1 Definition of packages

For the purpose of cohesion measurement, a package is defined as a set of elements (classes, interfaces or packages) and relations between elements. The presence of package within a package leads to the hierarchical organization of the package. A package at hierarchical level i can be represented as $p^i = \langle E^{i+1}, R^{i+1}\rangle$, where E^{i+1} represents the set of elements of a package p^i at level $i+1$, which may be classes, interfaces or packages, and R^{i+1} is a set of relations on E^{i+1} at hierarchical level $i+1$ i.e. $R^{i+1} \subseteq E^{i+1} \times E^{i+1}$. The relations on set of elements represent binary directed connections or relations between pairs of elements of the package. A package is used "to group elements and to provide a namespace for the grouped elements" [55].

Subpackage: For any package p^i in a system, $subP(p^i)$ denotes an element of p^i which is a package itself and is present at level $i+1$ in the hierarchy. A package $p_1^{i+1} = \langle E_1^{i+2}, R_1^{i+2}\rangle$ is said to be a subpackage of package $p_2^i = \langle E_2^{i+1}, R_2^{i+1}\rangle$ if $p_1^{i+1} \in E_2^{i+1}$ i.e., $p_1^{i+1} = subP(p_2^i)$. The top level in the package hierarchy in a software system consists of packages, which are not subpackages of any other package and the lowest level is occupied by the atomic members, which do not contain any subpackages.

Disjoint packages: All packages of a system are structured into a number of non-overlapping hierarchical levels, such that for any two packages p_1^i, p_2^i at the same level i, $E_1^{i+1} \cap E_2^{i+1} = \emptyset$. Then, packages p_1^i and p_2^i are said to be disjoint packages. Thus, packages defined at the same hierarchical level are always disjoint. This means, packages defined at the same level never share classes (or interfaces) among them. In other words, a single class (or interface) always belongs to a single package at a particular level of the package hierarchy.

Empty package: A package having no elements, and hence no relations, is termed an empty package. It is represented as $\langle\emptyset, \emptyset\rangle$.

3.2 Definitions of relations

The relation between a pair of elements of a package can be of the type inheritance relation, aggregation relation, reference relation, etc. [15]. This type of relation between a pair of elements e_1^i and e_2^i of a package present at hierarchical level i is represented by $r(e_1^i, e_2^i)$. The presence of a relation between these elements is also denoted by $r(e_1^i, e_2^i) = 1$ or $e_1^i \to e_2^i$. These relations are asymmetric in nature i.e. $e_1^i \to e_2^i$ does not imply that $e_2^i \to e_1^i$. The presence of different types of element in a package such as classes, interfaces or subpackages causes different types of relation among package elements. However, interfaces are quite similar to abstract classes in structure as well as in behavior, as far as package level cohesion measurement is concerned. Thus, in this paper, interfaces have been treated just as a special type of class. By following the above discussion, the possible types of relation among package elements are described as follows.

Class–Class relation: This type of relation exists between a pair of classes (or interfaces) of a package due to the presence of different types of relation between them such as aggregation, inheritance, reference relations etc. [15]. This relation between a pair of elements of a package, p^i ($p^i = \langle E^{i+1}, R^{i+1}\rangle$ where $r(e_1^{i+1}, e_2^{i+1}) \in R^{i+1}$ & $e_1^{i+1}, e_2^{i+1} \in E^{i+1}$) is represented as $r(e_1^{i+1}, e_2^{i+1}) = 1 | (r(c_1^{i+1}, c_2^{i+1}) = 1 \wedge c_1^{i+1} = e_1^{i+1} \wedge c_2^{i+1} = e_2^{i+1})$ or it can be said that if there is a relation between two classes of a package, then there exists a relation between package elements i.e. $c_1^{i+1} \to c_2^{i+1} \Rightarrow e_1^{i+1} \to e_2^{i+1}$, where $(c_1^{i+1} = e_1^{i+1} \wedge c_2^{i+1} = e_2^{i+1})$. The relations of the type Interface–Class, Class–Interface or Interface–Interface are covered in Class–Class relation as interfaces have been considered as a special type of class during package cohesion measurement.

Package–Package relation: This type of relation exists between two subpackages of a package. Such types of relation are recursively defined. One package is said to have a relation with the other package if elements of one package have got one or more relations with the elements of the other package. This type of relation between a pair of elements of a package p^i ($p^i = \langle E^{i+1}, R^{i+1}\rangle$ where $r(e_1^{i+1}, e_2^{i+1}) \in R^{i+1}$ & $e_1^{i+1}, e_2^{i+1} \in E^{i+1}$) is given as $r(e_1^{i+1}, e_2^{i+1}) = 1 | r(c_1^{\geq i+2}, c_2^{\geq i+2}) = 1 \wedge c_1^{\geq i+2} \in e_1^{i+1} \wedge c_2^{\geq i+2} \in e_2^{i+1} \wedge e_1^{i+1} = subP(p^i) \wedge e_2^{i+1} = subP(p^i)$. In other words, if classes (or interfaces) present at next or higher levels belonging to a pair of subpackage elements of a package are related, then these subpackage elements of package are also related i.e. $c_1^{\geq i+2} \to c_2^{\geq i+2} \Rightarrow e_1^{i+1} \to e_2^{i+1}$, where $(c_1^{\geq i+2} \in e_1^{i+1} \wedge c_2^{\geq i+2} \in e_2^{i+1} \wedge e_1^{i+1} = subP(p^i) \wedge e_2^{i+1} = subP(p^i))$.

Subpackage–Class relation: This type of relation originates from a subpackage to a class (or interface) of a package at the other end. A subpackage (at level $i+1$) of a package (at level i) is said to be related to a class (or interface) at level $i+1$ of the package, if any class or interface (present at $i+2$ or higher levels) of the subpackage has got a relation with the class or interface of the package. This type of relation in a package p^i ($p^i = \langle E^{i+1}, R^{i+1}\rangle$ where $r(e_1^{i+1}, e_2^{i+1}) \in R^{i+1}$ & $e_1^{i+1}, e_2^{i+1} \in E^{i+1}$) can be denoted as $r(e_1^{i+1}, e_2^{i+1}) = 1 | r(c_1^{\geq i+2}, c_2^{i+1}) = 1 \wedge c_1^{\geq i+2} \in e_1^{i+1} \wedge c_2^{i+1} = e_2^{i+1} \wedge e_1^{i+1} = subP(p^i) \wedge e_2^{i+1} \in E^{i+1}$. This type of relation can also be represented as $c_1^{\geq i+2} \to c_2^{i+1} \Rightarrow e_1^{i+1} \to e_2^{i+1}$, where $(c_1^{\geq i+2} \in e_1^{i+1} \wedge c_2^{i+1} = e_2^{i+1} \wedge e_1^{i+1} = subP(p^i) \wedge e_2^{i+1} \in E^{i+1})$.

Class–Subpackage relation: A Class–Subpackage relation exists due to relation from a class (or interface) to a subpackage of a package. The class is said to be connected to the subpackage, if there exists a relation from class (or interface) at level $i+1$ to any class (or interface) at

$i+2$ or higher levels of the subpackage (at level $i+1$) of the package p^i ($p^i = \langle E^{i+1}, R^{i+1}\rangle$ where $r(e_1^{i+1}, e_2^{i+1}) \in R^{i+1}$ & $e_1^{i+1}, e_2^{i+1} \in E^{i+1}$). This type of relation can be denoted as $r(e_1^{i+1}, e_2^{i+1}) = 1 | r(c_1^{i+1}, c_2^{\geq i+2}) = 1 \wedge c_1^{i+1} = e_1^{i+1} \wedge c_2^{\geq i+2} \in e_2^{i+1} \wedge e_1^{i+1} \in E^{i+1} \wedge e_2^{i+1} = subP(p^i)$. Alternatively, this type of relation can be specified as $c_1^{i+1} \rightarrow c_2^{\geq i+2} \Rightarrow e_1^{i+1} \rightarrow e_2^{i+1}$, where ($c_1^{i+1} = e_1^{i+1} \wedge c_2^{\geq i+2} \in e_2^{i+1} \wedge e_1^{i+1} \in E^{i+1} \wedge e_2^{i+1} = subP(p^i)$).

4 Package cohesion measurement

As defined above, a package is a set of classes, interfaces and subpackages leading to a hierarchical structure of the object-oriented system. At a particular hierarchical level, a package can be viewed as a set of elements (classes, interfaces or subpackages) and relations between pairs of these elements at the next hierarchical level. Different possible types of these relations have already been described in the previous section. These relations among elements of a package lead to the intra-package dependency in a package and this dependency among elements of a package can effectively represent cohesion of the package. During measurement of cohesion of a package, we consider only the cohesiveness among its elements (classes, interfaces or subpackages) present at next hierarchy level and do not take into account the individual cohesions of the elements. The count of above described relations among package elements determine how much cohesive a package is. The degree of cohesiveness among elements of a package is measured as the ratio of the actual number of relations between ordered and unique pairs of package elements and maximum possible number of relations between ordered and unique pairs of elements of a package. Thus, cohesion of a package p^i ($p^i = \langle E^{i+1}, R^{i+1}\rangle$) present at hierarchical level 'i' having n elements is defined as

$$PCoh(p^i) = \begin{cases} 0 & \text{if } (n=0) \\ \frac{\sum_{x=1}^{n}\sum_{y=1 \wedge y\neq x}^{n-1} r(ex^{i+1}, ey^{i+1})}{n(n-1)} & \text{if } (n>1) \\ 1 & \text{if } (n=1) \end{cases}$$

where n is the number of elements (classes, interfaces or subpackages) at hierarchical level $i+1$ in a package p^i ($p^i = \langle E^{i+1}, R^{i+1}\rangle$) defined at level i and e_x^{i+1} and e_y^{i+1} together make a pair of elements of package p^i. As discussed earlier, relations between these elements is denoted by $r(e_x^{i+1}, e_y^{i+1})$, where $r(e_x^{i+1}, e_y^{i+1}) \in R^{i+1}$ and $e_x^{i+1}, e_y^{i+1} \in E^{i+1}$. Also, $r(ex^{i+1}, ey^{i+1})$represents a relation (as defined above) between a pair of package elements. Two package elements are said to be related only if there exists at least one relation between them and the relations can be one of the types as explained in the previous section. $\sum_{x=1}^{n}\sum_{y=1 \wedge y\neq x}^{n-1} r(ex^{i+1}, ey^{i+1})$ represents number of binary directed relations (at hierarchical level $i+1$) between all ordered and unique pairs of n elements of the package p^i, where $r(ex^{i+1}, ey^{i+1})$ denotes a relation between a pair of package elements at level $i+1$ and also in accordance with the above discussions, $r(ex^{i+1}, ey^{i+1})$ may or may not be equal to $r(ey^{i+1}, ex^{i+1})$. Further, cohesion of a package p^i ($p^i = \langle E^{i+1}, \mathrm{R}^{i+1}\rangle$), $PCoh(p^i)$ has been defined in terms of relations between its elements i.e. $e_x^{i+1} \in E^{i+1}$ and $e_y^{i+1} \in E^{i+1}$. Thus, e_x^{i+1} and e_y^{i+1} are siblings present at the same hierarchical level. Hence, for calculating cohesion of a package, elements of a package which are siblings and present only at immediate next hierarchical level are considered and belong to the same tree structure.

There are a total of n elements in a package and each element of the package may be at the most connected to all other $n-1$ elements of the package. Thus the maximum possible number of relations among n elements is $n^*(n-1)$, which represents count of maximum possible intra-package dependency for a package. The ratio of the number of binary relations and the maximum possible number of binary relations gives us a normalized value for the proposed measure [18]. The cohesion value for a package will be 0, if there is no relation among its elements and 1 if every element of the package is related to all other elements. Here 0 represents the minimum and 1 indicates the maximum cohesion. The value of our proposed measure will always lie between a baseline value, 0 and a ceiling value, 1. This normalized value of the measure makes the proposed package cohesion metric independent of the size of the package for which cohesion is to be measured and allows meaningful comparisons between the cohesions of the packages of different sizes.

Other cases: If $n=0$, there is no element and hence no relation is possible in package p^i. Thus, package p^i is an empty package$\langle \emptyset, \emptyset\rangle$and the value of cohesion measure for an empty package is always zero. Thus, $PCoh(p^i) = 0$.

If $n=1$, it means package p^i contains a single element then, cohesion of package p^i with only one element is 1, because it contains all the relations it can possibly contain i.e. $PCoh(p^i) = 1$.

Relation between package cohesion and package elements coupling: Packages should be designed with the objective of high level of cohesion and a low level of coupling between them. If we observe inside the package: relations among package elements are modeled via coupling between package elements (classes or interfaces). Two package elements are coupled when they have some relation between them. If we observe the package from outside: relations among package elements are modeled via package cohesion.

Illustration of package cohesion measurement:

```
package myshapes;

public interface Drawable {
      public void draw(Graphics g);
      }

class Line implements Drawable {
      public void draw(Graphics g) {
         . . . // do something – presumably,
draw a line
      }
      . . . // other methods and variables
   }
```

```
package myshapes.round;
import myshapes.Drawable;
public class Circle {
   public void findArea( ) {
         . . . // find area of circle
         }
   . . . // other methods and variables
   }
Class FilledCircle extends Circle implements
Drawable{
      public void draw(Graphics g) {
         . . . // do something – draw a filled Circle
            }
      void findArea( ) {
         . . . // find area of filled circle
         }
         . . . // other methods and variables
      }
```

A package is cohesive when its elements have high number of relations among them. Thus, cohesion of a package can be defined in terms of coupling among elements of the package.

The above example of code can be represented using a UML diagram [55] as shown in Fig. 1. The relations in this diagram are shown as directed arrows from one element to the other.

According to the definitions of packages in Sect. 3, package 'myshapes' has got three elements and two relations between different pairs of its elements. These elements include interface 'Drawable', class 'Line' and subpackage 'round'. There exists one relation from class 'Line' to interface 'Drawable', because class 'Line' implements interface 'Drawable' and the other relation is between subpackage 'round' and interface 'Drawable' due to the existence of a relation between class-element 'FilledCircle' of subpackage 'round' and interface 'Drawable'. If package 'myshapes' is supposed to be at the hierarchical level 0, then its elements are at hierarchical level 1. Thus, package 'round' is at level 1 and its elements and relations among its elements are at level 2, which are encapsulated within the package 'round' and are not visible outside. Thus, during measurement of cohesion of a package 'myshapes', the relation between 'FilledCircle' (element of subpackage 'round') and interface 'Drawable' (element of package 'myshapes') is considered as the relation between subpackage 'round' and interface 'Drawable'.

According to the definition of the measure, $n = 3$ and $\sum_{x=1}^{n} \sum_{y=1 \wedge y \neq x}^{n} r(ex^{i+1}, ey^{i+1}) = 2$.

Thus, Cohesion of the package 'myshapes' at hierarchical level 0 is given by the proposed measure as

$$PCoh(\text{myshapes}^0) = \frac{2}{3 * (3-1)} = 0.334.$$

5 Theoretical validation of package cohesion measure

The purpose of this section is to provide the theoretical soundness to the proposed measure, i.e., the fact that it really measures the software characteristic it is supposed to measure, which is an obvious prerequisite for its acceptability and use [13, 67]. The proposed measure is evaluated theoretically by using the four properties given by Briand et al. [13]. We reviewed different validation frameworks available in literature before choosing Briand et al. validation framework. Weyuker's framework [68] has been defined and mainly used for evaluation of complexity measures. Some other evaluation frameworks such as Zuse framework [75] and Tian & Zelkowitz [66] axioms are also used for validation of complexity measures. The four cohesion properties defined by Briand et al. framework are one of the more recent proposals to characterize cohesion in a reasonably intuitive and rigorous manner. While these properties are not sufficient to say that a measure which fulfills them all will be useful, it is likely that a measure which does not fulfill them all is ill-defined.

Property 1 *Non-Negativity and Normalization.*

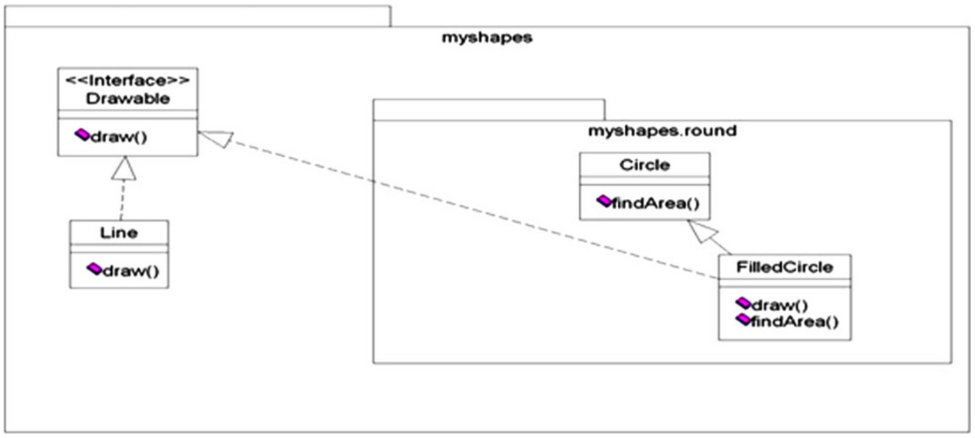

Fig. 1 Graphical representation for the above example

According to the above given definition of the proposed measure, cohesion of a package p^i in an object-oriented system belongs to a specified interval i.e. $PCoh(p^i) \in [0, 1]$. Thus, the value of package cohesion given by the proposed measure will always be non-negative and normalized.

Property 2 *Null Value and Maximum Value.*

As discussed above, if a package, $p^i = \langle E^{i+1}, R^{i+1}\rangle$ is empty i.e. $p^i = \langle \emptyset, \emptyset\rangle$ means if there is no element and no relation within a package i.e. $E^{i+1} = \emptyset$ and $R^{i+1} = \emptyset$, then cohesion of package p^i, $PCoh(p^i) =$ null i.e. $p^i = \langle \emptyset, \emptyset\rangle \Rightarrow PCoh(p^i) = 0$ (Null value).

Cohesion of a package, $p^i = \langle E^{i+1}, R^{i+1}\rangle$ where number of elements, E^{i+1} is n and number of relations, R^{i+1} is $n * (n - 1)$ i.e. each element of package p^i is connected to each other element of the package, then $PCoh(p^i) = 1$ or in case package p^i contains single element ($n = 1$), then as discussed above, $PCoh(p^i) = 1$ i.e. $n = 1$ or $R^{i+1} = n * (n - 1) \Rightarrow PCoh(p^i) = 1$ (maximum value).

The proposed measure has got the null value and the maximum value of cohesion for a package in well defined situations. Thus, the proposed measure satisfies Property 2.

Property 3 *Monotonicity.*

As per Briand et al.'s framework, cohesion Property 3 of monotonicity requires that adding relationships must not decrease cohesion [13]. This property requires addition of relations not the elements to prove its compliance.

Let p be a package with relations R in an object-oriented system. Package p is modified to form a new package p' with relations R', which is identical to p except that $R \subset R'$, i.e., some relations are added in p. Then, according to the above given definition of the measure, the numerator value will only increase or will remain the same but will never decrease.

For package $p = \langle E, R\rangle$ and $p' = \langle E, R'\rangle$.

If $R \subset R'$ then $PCoh(p) \le PCoh(p')$.

Thus, the proposed measure satisfies this property as well. The additional internal relations in a package will never decrease its cohesion.

Property 4 *Merging of Unconnected Packages.*

This property states that merging of two unconnected packages must not increase cohesion of the resulting package. When two or more packages having no relations between them are merged, cohesion should not increase because apparently unrelated elements are being encapsulated together in a single package.

Let p_1 and p_2 be two packages in an object-oriented system S. Let p' be the package, which is the union of p_1 and p_2. Let S' be the object-oriented system, which is identical to S except that packages p_1 and p_2 are replaced by p'. If no relations exist between packages p_1 and p_2, then according to the above given definition of the measure, increase in value of the numerator will be less in comparison to the value of the denominator as unified package will contain elements as well as relations of both packages. Thus, cohesion of unified package may decrease i.e. $\max\{PCoh(p_1), PCoh(p_2)\} \ge PCoh(p')$.

Hence, the proposed measure satisfies this property very well.

6 Experimental validation of proposed measure

The ISO/IEC 9126-1 standard [41] on software product quality states that internal product measures should be related to external quality attributes in order to be useful and meaningful. In this section, we attempt to assess experimentally whether the proposed package cohesion metric is a useful indicator of external quality attribute of packages such as reusability. This would help us to evaluate the proposed package cohesion metric as a quality indicator.

6.1 Experiment goal

The goal of this experimental study is to analyze experimentally the proposed metric for the purpose of evaluating whether or not this metric is useful for indicating the reusability of the packages. We used the Goal/Question/Metric (GQM) paradigm [6, 7, 17, 62] which provide a template as well as guidelines to define measurement goals in a systematic manner. The measurement goal of our experimental study is defined as follows:

- Object of study: package.
- Purpose: analysis.
- Quality focus: effort required to reuse a package (Package Reusability).
- Viewpoint: software developer.
- Environment: open-source software projects developed in Java.

These five goal dimensions have a direct impact on the remaining steps of the experimental validation of the measure [17]. The object of study helps to determine the software artifacts that must be modeled so that they are analyzable. It also helps in defining the hypotheses that may be relevant because they are directly related to the object of study. The purpose helps to determine the type and amount of data to be collected. The quality focus facilitates in determining the dependent attribute(s) against which the defined metric is going to be experimentally evaluated. The viewpoint assists to determine the point in time at which analysis should be carried out. The environment helps to determine the context in which the study is being carried out.

6.2 Empirical hypotheses

An empirical hypothesis is a statement believed to be true about the relation between one or more attributes of the object of study and the quality focus [17]. In this case, the hypotheses are about the relationship between cohesion of a package (object of study) and reusability of the package (quality focus).

The analysis is based on the hypotheses:

$H_0 : p = 0$ (Null hypothesis)—There is no significant correlation between the proposed package cohesion metric and package reusability.

$H_1 : p \neq 0$ (Alternative hypothesis)—There is significant correlation between the proposed package cohesion metric and package reusability.

6.3 Experimental environment

Open-source software projects are widely available on web for both use and research. These projects offer a diverse set of solutions which encourage a collaborative, consensus-based development process under open software licenses. The sample used for this experimental study was taken from six open-source software projects whose source code was readily available for use. The major reason behind selection of these projects was that these software projects were developed in Java and were organized using packages. The presence of packages in these projects made it possible to apply package level metrics on them. Twenty-five packages taken from six open-source projects were used in order to experimentally evaluate the proposed metric. Out of these six projects, four belonged to the Apache software foundation and eighteen packages belonging to these four projects were downloaded from Apache Jakarta website [65]. The Apache project Byte Code Engineering Library (BCEL) is intended to give users a convenient possibility to analyze, create, and manipulate Java class files, and classes are represented by objects which contain all the symbolic information of the given class like methods, fields and byte code instructions [22]. The Apache project Bean Scripting Framework (BSF) is a set of Java classes which provides scripting language support within Java applications, and access to Java objects and methods from scripting languages. BSF allows one to write JSPs in languages other than Java while providing access to the Java class library [9]. The Apache project Jakarta-ORO is a set of text-processing Java classes that provide Perl5 compatible regular expressions, AWK-like regular expressions, glob expressions, and utility classes for performing substitutions, splits, filtering filenames, etc. [42]. The Apache project Element Construction Set (ECS) is a Java API for generating elements for various markup languages such as HTML 4.0 and XML and can easily be extended to create tags for any markup language [33]. The XGen Source Code Generator [69] creates a Java source code from a simple XML document. Its primary function is to generate JDBC compliant beans that allow object level persistence to relational databases. The other project used in this study is JUnit [43] which is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. Table 1 shows the projects and the corresponding packages used from these projects for experimental study in this work. This table also provides general details about each package used in the experimental study in terms of size (in LOC) and the total number of classes contained in the package.

6.4 Analysis methodology

A statistical analysis is performed to correlate the proposed package cohesion metric with package reusability i.e. effort required to reuse a package. Correlation is a common

Table 1 Description of packages used in the study

Sr. No.	Project name	Package name	Size (LOC)	Total No. of classes
1	Byte Code Engineering Library (BCEL)	org.apache.bcel.verifier	12244	48
2		org.apache.bcel.verifier.exc	641	14
3		org.apache.bcel.verifier.statics	3425	9
4		org.apache.bcel.verifier.structurals	6289	14
5		org.apache.bcel.util	3906	20
6	Bean Scripting Framework (BSF)	org.apache.bsf.util.event	1790	21
7		org.apache.bsf.util.event.generator	1110	4
8		org.apache.bsf.util	5835	54
9		org.apache.bsf.util.type	239	2
10		org.apache.bsf.util.cf	570	2
11	Jakarta-ORO	org.apache.oro.io	494	4
12		org.apache.oro.text	14715	50
13		org.apache.oro.text.awk	3383	17
14		org.apache.oro.text.perl	1477	3
15		org.apache.oro.util	991	7
16	Element Construction Set (ECS)	org.apache.ecs.jsp	1834	15
17		org.apache.ecs.storage	452	3
18		org.apache.ecs.xml	786	3
19	XGen Source Code Generator	workzen.xgen.ant.legacy	1193	4
20		workzen.xgen.engine	426	2
21		workzen.xgen.loader	1009	7
22	JUnit	junit.samples	541	2
23		junit.samples.money	390	4
24		junit.tests	1583	36
25		junit.tests.extensions	248	5

research method for empirically validation of the metrics. For example, Kabaili et al. [44] used this statistical method to evaluate cohesion metrics as changeability indicators. Mitchell and Power [53] used this method to examine the usefulness of the run time coupling metrics. Li and Henry [48] examined correlations between various metrics in order to determine their usefulness.

Correlation is measured using the standard Karl Pearson Product-Moment correlation coefficient r, which measures the degree and direction of the linear relationship between the two variables. Karl Pearson's coefficient is the most widely used method of measuring correlation. This method assumes that there is a linear relationship between two variables and one of the variables is independent while the other is dependent. Karl Pearson's coefficient of correlation is given by [45]:

$$r = \frac{\sum(Xi - \overline{X})(Yi - \overline{Y})}{n * \sigma x * \sigma y}$$

where X_i = ith value of X variable, Y_i = ith value of Y variable, $\overline{X}$ = mean of X, $\overline{Y}$ = mean of Y, n = number of pairs of observations of X and Y, σ_x = standard deviation of X, σ_y = standard deviation of Y.

The value of correlation coefficient (r) lies between -1 and $+1$ through 0, where 1 represents a perfect positive correlation between the variables; -1 denotes a perfect negative correlation; and 0 indicates that there is no linear relationship between the variables. The degree of the correlation is determined by the magnitude of the coefficient. Adjective ratings of correlation strength follow the definitions developed by Cohen [27]:

- ⟨0.1 “Trivial”
- 0.1 to 0.3 “Minor”
- 0.3 to 0.5 “Moderate”
- 0.5 to 0.7 “Large”
- 0.7 to 0.9 “Very large”
- 0.9 to 1 “Almost perfect”.

Table 2 Package cohesion measurement

Sr. No.	Package name	No. of elements	No. of relations	Package cohesion metric (*PCoh*)
1	org.apache.bcel.verifier	14	56	0.30
2	org.apache.bcel.verifier.exc	14	10	0.15
3	org.apache.bcel.verifier.statics	9	1	0.013
4	org.apache.bcel.verifier.structurals	14	10	0.054
5	org.apache.bcel.util	20	10	0.026
6	org.apache.bsf.util.event	6	18	0.60
7	org.apache.bsf.util.event.generator	4	4	0.33
8	org.apache.bsf.util	20	30	0.078
9	org.apache.bsf.util.type	2	1	0.50
10	org.apache.bsf.util.cf	2	1	0.50
11	org.apache.oro.io	4	3	0.25
12	org.apache.oro.text	15	34	0.16
13	org.apache.oro.text.awk	17	41	0.15
14	org.apache.oro.text.perl	3	0	0
15	org.apache.oro.util	7	11	0.26
16	org.apache.ecs.jsp	15	14	0.067
17	org.apache.ecs.storage	3	3	0.50
18	org.apache.ecs.xml	3	2	0.33
19	workzen.xgen.ant.legacy	4	3	0.25
20	workzen.xgen.engine	2	1	0.50
21	workzen.xgen.loader	7	7	0.167
22	junit.samples	4	1	0.08
23	junit.samples.money	4	9	0.75
24	junit.tests	5	4	0.20
25	junit.tests.extensions	5	4	0.20

Any relationship between two variables should be assessed for its strength as well as its significance. The significance of the correlation results is assessed by the p-value. The p-value corresponds to the probability that the measured correlation could be due to purely random effects. The smaller the p-level, the more significant is the relationship between the variables.

However, correlation study can only establish a relationship between two variables (in our case, package cohesion and package reusability) but could not establish a cause-effect relationship between the two.

6.5 Analysis of experimental results

Table 2 shows the number of elements (present at the next hierarchical level) and the number of relations among elements of each package and the corresponding values of the package cohesion metric (*PCoh*) for each package under study.

In Fig. 2, we show the measured values of the Package cohesion metric for all 25 packages graphically. In this figure, Packages Sr. Nos. as given in Table 2 are taken on the X-axis and their corresponding package cohesion values are plotted on the Y-axis.

Table 3 Descriptive statistics of the analyzed package cohesion metric

Statistical parameter	Package cohesion metric (*PCoh*)
Maximum value	0.75
Minimum value	0
Median	0.20
Mean	0.26
Standard deviation	0.20

Table 3 provides common descriptive statistics of the metric distributions across all the packages used in the experimental study. This table shows that package cohesion metric (*PCoh*) has enough variance to distinguish the packages. Thus, the proposed measure is effective to be used in analysis procedure.

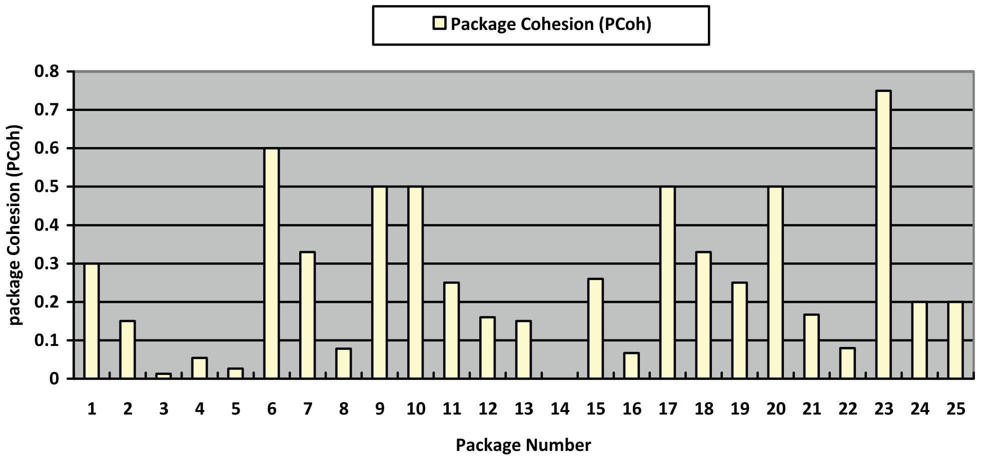

Fig. 2 Package cohesion values for all packages

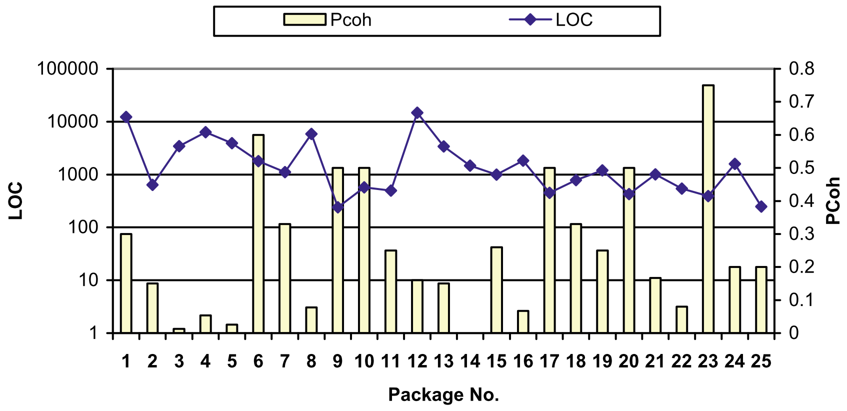

Fig. 3 Relationship between *PCoh* and LOC

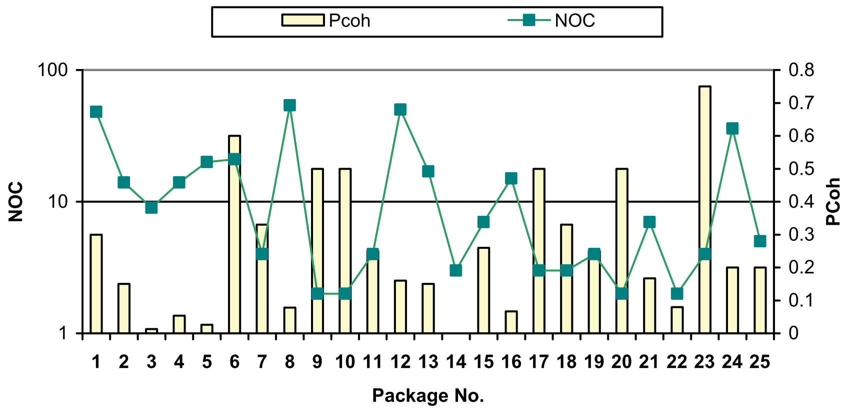

Fig. 4 Relationship between *PCoh* and No. of classes (NOC)

6.5.1 Relationship between package cohesion and package size

Figure 3 shows the relationship between package cohesion (*PCoh*) and package size in lines of code (LOC) for all 25 packages under study.

Figure 4 shows the relationship between cohesion of a package (*PCoh*) and number of classes (NOC) in the package.

From Figs. 3 and 4, it can be easily observed that the cohesion value measured by the proposed package cohesion metric (*PCoh*) has no association with the size of the corresponding package measured in LOC and the number of classes in the package. To strengthen the above observation, correlation coefficients are computed between package cohesion (*PCoh*) & LOC and *PCoh* & NOC for all 25 packages. For calculation of the correlation coefficients, the Karl Pearson Product-Moment correlation method [27, 45] is used. The value of correlation coefficient between *PCoh* and LOC for 25 packages comes out to be 0.27 and correlation coefficient between *PCoh* & NOC is 0.35 for 25 pack-

Table 4 Ratings of reusability assigned to each package

Sr. No.	Package name	Rating of package reusability
1	org.apache.bcel.verifier	5
2	org.apache.bcel.verifier.exc	2
3	org.apache.bcel.verifier.statics	2
4	org.apache.bcel.verifier.structurals	2
5	org.apache.bcel.util	1
6	org.apache.bsf.util.event	8
7	org.apache.bsf.util.event.generator	6
8	org.apache.bsf.util	2
9	org.apache.bsf.util.type	8
10	org.apache.bsf.util.cf	7
11	org.apache.oro.io	9
12	org.apache.oro.text	2
13	org.apache.oro.text.awk	2
14	org.apache.oro.text.perl	1
15	org.apache.oro.util	2
16	org.apache.ecs.jsp	2
17	org.apache.ecs.storage	7
18	org.apache.ecs.xml	6
19	workzen.xgen.ant.legacy	4
20	workzen.xgen.engine	8
21	workzen.xgen.loader	3
22	junit.samples	1
23	junit.samples.money	7
24	junit.tests	4
25	junit.tests.extensions	3

ages. These low values of correlation coefficients indicate that the proposed package cohesion measure (*PCoh*) is independent of the size of the package (in terms of LOC as well as NOC) for which cohesion is being measured. Thus, the proposed measure (*PCoh*) allows meaningful comparisons between cohesion values of different packages of different sizes.

6.5.2 *Relationship between package cohesion and package reusability*

An evaluation team consisting of five software developers was assigned the task of evaluating the reusability of 25 packages. These software developers were computer engineering graduates working in my organization. All software developers had about three years of work experience in commercial software development, and have been working on development of many Java-based projects. The team of developers was required to use each package to complete an unfinished application. The task of each developer was to choose the required classes from a given package and integrate them to the application. The team was free to modify these packages to reuse them. The team assessed the effort required to reuse the package and rated the reusability of each package on a numerical scale from 1 (low) to 10 (high). The more the effort required reusing a package, the less is the reusability of the package. A package with low reusability rating such as 1 means the more effort required to reuse the package, and a package with high rating such as 10 means the less effort required to reuse the package. To rate the reusability of a package, the team of developers considered the following effort in reusing it:

- The effort to select the right classes from the package for reuse.
- The effort to modify the reused classes if needed.
- The effort to find and integrate all required classes.
- The effort to test the correctness of the integration.

Table 4 shows the ratings of reusability of each package assigned by the team of software developers depending on the effort required to reuse the corresponding package.

Figure 5 shows relationships between measured values of package cohesion (as shown in Table 2) and package reusability ratings (as depicted in Table 4) for each package. It can easily be seen that the package cohesion metric

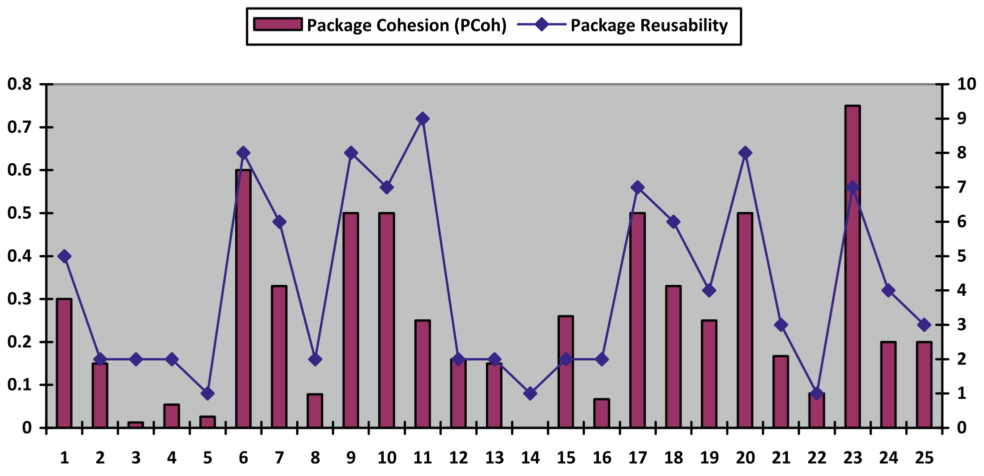

Fig. 5 Relationship between package cohesion & package reusability

Table 5 Correlation values for cohesion measures

	PCoh	LCOM	LCOM*	TCC	CAMC	RC	EEC	ICH	SCC
Correlation coefficient	0.69	−0.32	−0.34	0.41	0.29	0.31	0.43	0.12	0.27
Significance level	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

(*PCoh*) values have a strong positive correlation with the package reusability ratings.

As stated above, the Karl Pearson Product-Moment correlation method [27, 45] was used to quantify the correlation between the proposed package cohesion metric and package reusability as rated by the team of software developers. The computed value of the correlation coefficient (r) between the proposed metric and team's reusability ratings comes out to be 0.69 at the significance level 0.05. Such a high value of correlation coefficient shows a strong positive correlation between the proposed package cohesion metric and the package reusability. Therefore, we reject the null hypothesis and accept the alternative hypothesis (as stated above) and conclude that there is a significant correlation between the proposed package cohesion metric and package reusability. The strong correlation between the proposed metric and reusability signify that the proposed package cohesion metric is a good indicator of the external quality attributes of the package.

6.6 Comparison of proposed measure with existing measures

The proposed metric was compared experimentally with the related cohesion metrics reviewed in Sect. 2 using 25 packages listed above. For this purpose, we used Karl Pearson correlation method [27, 45] to compute the correlation coefficients between the package reusability ratings given by team of developers (as given in Table 4) and the values produced by the various cohesion measures such as LCOM [25, 26], LCOM* [39], TCC [10], CAMC [4, 5], RC [51], EEC [1], ICH [47], SCC [74]. The Lack of Cohesion in Methods (LCOM) metric [25, 26] was proposed by Chidamber and Kemerer to measure lack of cohesion in a class and LCOM* [39] metric was an improved version of LCOM [25, 26] given by Henderson-Sellers. The Tight Class Cohesion (TCC) metric proposed by Bieman and Kang measures the percentage of pairs of the public methods of the class which have direct common attribute usage [10]. The Cohesion Among Methods of Classes (CAMC) metric was proposed by Bansiya et al., which evaluated the relatedness of methods in the interface of a class using the parameter lists defined for the methods [4, 5]. The Relational Cohesion (RC) metric proposed by Martin is defined as the ratio of the number of data relations in a package to the number of components in the package [51]. Allen et al. proposed an information-theory-based metric named excess entropy cohesion (EEC) metric [1]. Lee and Liang proposed a cohesion measure, Information-based cohesion, (ICH) for a set of classes based on information flow through method invocations within classes [47]. They defined cohesion of a set of classes as the sum of the cohesion of the classes in the set. Zhou et al. [74] proposed a measure called SCC for measuring semantic cohesion of a package by considering the fact that two components of a package are related tightly if they have similar contexts.

The results of this correlation study are given in Table 5.

From the results of the above study, it can easily be observed that the proposed measure (*PCoh*) produced the highest magnitude of correlation with the package reusability among the various cohesion measures. Some of the cohesion measures such as LCOM [25, 26] and LCOM* [39] produced negative correlation coefficients as these metrics measure lack of cohesion. These results suggest that the proposed package level cohesion measure is a better indicator of reusability than the other existing cohesion measures.

6.7 Threats to validity

As with other empirical experiments, there are threats to the validity of this study. The build threat is the reuse effort may not represent the quality of the package. The quality of a package is dependent on many other factors such as understandability, maintainability. To reduce this threat in the future, other measures of quality such as maintenance effort could be used to validate the proposed metric. The internal threat is the measurement of reuse effort. In our study, a team of five developers worked on all packages. The number of developers taking part in the experiment is very small which is a threat to validity of the experiment and the ratings of package reusability may depend on expertise and prior experience of software developers which is also a major threat to its validity. To reduce these threats, various teams having a specific number of software developers can be constructed and then average value of the ratings assigned by these different teams can be taken to reduce the bias being caused by the personal experiences of the software developers. The threats to external validity mainly include the subject software as these packages may not be representative of other software products. Also, the number of packages (25 no. packages) taken from open-source software used in the current study is too small which is a threat to the validity of the experiment. To reduce these threats, a large set of packages can be used in future studies and more studies can be conducted using commercial software systems. Another threat to validity of this study is that correlation analysis has been performed between package cohesion and package reusability. However, correlation study is able to establish relationship between package cohesion and package reusability but could not establish cause-effect relationship between the two. Further, correlation study performed in the paper requires a number of assumptions such as normality, linearity, and homoscedasticity need to be verified.

7 Conclusions and future work

In this paper, we have proposed a new approach to measure the cohesion of packages based on formal definitions and properties of the packages. The proposed measures are based on the relations present between pairs of package elements, which can be classes, interfaces or subpackages. The cohesion of a package is measured as the number of binary directed relations between all ordered and unique pairs of the package elements, divided by the maximum possible number of relations between them. The hierarchical structure of packages has also been taken into consideration during the measurement of package cohesion. The proposed measure is validated theoretically with the help of the four well-established properties given by Briand et al. [13] and an experimental study has been conducted using 25 packages taken from six open-source software projects developed in Java. The usefulness of proposed measure is proved by empirically validating the proposed measure as the indicator of external quality attribute of packages such as reusability. The proposed measure for package cohesion is also a better predictor of code reusability than the existing cohesion measures as shown in the experimental study.

The proposed package level cohesion measurement process described in this research is an important step toward designing high quality software. We believe that the study presented herein should encourage other researchers and developers to adapt and use this metric in combination with other design principles to develop more package level metrics for the measurement of other software quality attributes. In addition, there is need of further empirical investigations of the proposed package level metric in order to establish its relations with other external software quality factors such as maintainability, adaptability.

References

1. Allen E, Khoshgoftaar T, Chen Y (2001) Measuring coupling and cohesion of software modules: an information theory approach. In: Proceedings of the seventh international software metrics symposium, 2001
2. Anquetil N, Lethbridge T (1999) Experiments with clustering as a software remodularization method. In: Proceedings of WCRE '99 (6th working conference on reverse engineering), Louis Pasteur, University of Ottawa, Ottawa, Canada, 1999, pp 235–255
3. Atole CS, Kale KV (2006) Assessment of package cohesion and coupling principles for predicting the quality of object oriented design. In: Proceedings of the 1st international conference on digital information management, Dec 2006, pp 1–5
4. Bansiya J, Davis C (1999) Class cohesion metric for object-oriented designs. J Object-Oriented Program 11(8):47–52
5. Bansiya J, Davis C (2002) A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng 28(1):4–17
6. Basili VR, Weiss D (1984) A methodology for collecting valid software engineering data. IEEE Trans Softw Eng 10(11):728–738
7. Basili VR, Rombach DH (1988) The tame project: towards improvement-oriented software environments. IEEE Trans Softw Eng 14(6):758–773
8. Basili VR (1997) Evolving and packaging reading technologies. J Syst Softw 38(1):3–12
9. Bean Scripting Framework (BSF) (2011) http://jakarta.apache.org/bsf/index.html

10. Bieman JM, Kang BK (1995) Cohesion and reuse in an object-oriented system. In: Proceedings of the symposium on software reusability (SSR'95), Seattle, WA, 1995, pp 259–262
11. Bieman JM, Kang BK (1998) Measuring design-level cohesion. IEEE Trans Softw Eng 24(2):111–124
12. Bieman JM, Ott LM (1994) Measuring functional cohesion. IEEE Trans Softw Eng 20(8):644–658
13. Briand L, Morasca S, Basili V (1996) Property-based software engineering measurement. IEEE Trans Softw Eng 22(1):68–86
14. Briand LC, Daly JW, Wust J (1998) A unified framework for cohesion measurement in object-oriented systems. Empir Softw Eng 3(1):65–117
15. Briand LC, Daly JW, Wust JK (1999) A unified framework for coupling measurement in object-oriented systems. IEEE Trans Softw Eng 25(1):91–121
16. Briand LC, Morasca S, Basili VR (1999) Defining and validating measures for object-based high-level design. IEEE Trans Softw Eng 25(5):722–743
17. Briand L, Morasca S, Basili V (2002) An operational process for goal-driven definition of measures. IEEE Trans Softw Eng 28(12):1106–1125
18. Brito e Abreu F (1995) The MOOD metrics set. In: Proceedings of the ninth European conf object-oriented programming workshop metrics workshop on metrics, Aarhus, Denmark, 1995
19. Brito e Abreu F, Goulão M (2001) Coupling and cohesion as modularization drivers: are we being over-persuaded? In: Proceedings of the 5th European conference on software maintenance and reengineering (CSMR'2001), Lisboa, Portugal, March 2001. IEEE Computer Society Press, Los Alamitos
20. Bryton S (2008) Modularity improvements with aspect-oriented programming. MSc Thesis (F. Brito e Abreu, supervisor), FCT/UNL
21. Bryton S, Brito e Abreu F (2007) Towards paradigm-independent software assessment. In: Proceedings of 6th international conference on the quality of information and communications technology (QUATIC), 2007. IEEE Computer Society Press, Los Alamitos
22. Byte Code Engineering Library (BCEL) (2011) http://jakarta.apache.org/bcel/index.html
23. Chae HS, Kwon YR, Bae DH (2000) A cohesion measure for object-oriented classes. Softw Pract Exp 30(12):1405–1431
24. Chae HS, Kwon YR, Bae DH (2004) Improving cohesion metrics for classes by considering dependent instance variables. IEEE Trans Softw Eng 30(11):826–832
25. Chidamber SR, Kemerer CF (1991) Towards a metrics suite for object oriented design. In: Proceedings of OOPSLA'91, ACM SIGPLAN Notices, 26 Nov 1991, pp 197–211
26. Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. IEEE Trans Softw Eng 20(6):476–493
27. Cohen J (1988) Statistical power analysis for the behavioral sciences, 2nd edn. Lawrence Erlbaum, Mahwah
28. Corbi TA (1989) Program understanding: challenge for the 1990's. IBM Syst J 28(2):294–306
29. Counsell S, Mendes E, Swift S (2002) Comprehension of object-oriented software cohesion: the empirical quagmire. In: Proceedings of the 10th international workshop on program comprehension, 2002, pp 33–42
30. Counsell S, Swift S (2006) The interpretation and utility of three cohesion metrics for object-oriented design. ACM Trans Softw Eng Methodol 15(2):123–149
31. DeMarco T (1982) Controlling software projects. Yourdon Press, New York
32. Eder J, Kappel G, Schre M (1992) Coupling and cohesion in object-oriented systems. In: Proceedings of the conference on information and knowledge. ACM Press, New York
33. Element Construction Set (ECS) (2011) http://jakarta.apache.org/ecs/index.html
34. Emerson T (1984) A discriminant metric for module cohesion. In: Proceedings of the 7th international conference on software engineering (ICSE), 1984
35. Fenton N, Pfleeger SL (1996) Software metrics: a rigorous and practical approach, 2nd edn. International Thomson Computer Press, London
36. Gélinas JF, Badri M, Badri L (2006) A cohesion measure for aspects, J Object Technol 5:97–114
37. Gosling J, Yellin F (1996) The Java application programming interface, Vol #1 (Core packages)/#2 (window toolkit and applets). Addison-Wesley, Reading
38. Gui G, Scott PD (2006) Coupling and cohesion measures for evaluation of component reusability. In: Proceedings of the international workshop on mining software repositories, Shanghai, China, 2006, pp 18–21
39. Henderson-Sellers B (1996) Object-oriented metrics: measures of complexity. Prentice-Hall, New York
40. Hitz M, Montazeri B (1995) Measuring coupling and cohesion in object-oriented systems. In: Proceedings of the third international symposium on applied corporate computing (ISACC'95), Monterrey, Mexico, 1995, pp 25–27
41. ISO/IEC 9126-1 (2000) Software product quality, Part 1: Quality model
42. Jakarta-ORO (2011) http://jakarta.apache.org/oro/index.html
43. JUnit (2011) http://sourceforge.net/projects/junit/
44. Kabaili H, Keller R, Lustman F (2001) Cohesion as changeability indicator in object-oriented systems. In: Proceedings of IEEE conference on software maintenance and reengineering (CSRM), 2001, pp 39–46
45. Kothari CR (2007) Research methodology: methods & techniques, revised second edn. New Age International, New Delhi, pp 139–141
46. Kumar A, Kumar R, Grover PS (2008) Towards a unified framework for cohesion measurement in aspect-oriented systems. In: Proceedings of the 19th Australian conference on software engineering (ASWEC '08), 2008
47. Lee YS, Liang BS (1995) Measuring the coupling and cohesion of an object-oriented program based on information flow. In: Proceedings of the international conference on software quality, Maribor, Slovenia, 1995, pp 81–90
48. Li W, Henry S (1993) Object oriented metrics that predict maintainability. J Syst Softw 23(2):111–122
49. Lientz B, Swanson B (1980) Software maintenance management. Addison-Wesley, Boston
50. Mancoridis S, Mitchell BS, Chen Y, Gansner ER (1999) Bunch: a clustering tool for the recovery and maintenance of software system structures. In: Proceedings of ICSM '99 (international conference on software maintenance), Oxford, England, 1999
51. Martin R (2002) Agile software development, principles, patterns, and practices. Prentice-Hall, New York
52. Mišić VB (2001) Cohesion is structural, coherence is functional: Different views, different measures. In: Proceedings of the seventh international software metrics symposium (METRICS-01), 2001. IEEE Press, New York
53. Mitchell A, Power J (2004) An empirical investigation into the dimensions of run time coupling in Java programs. In: Proceedings of the 3rd international conference on principles and programming in Java (PPPJ'04), Las Vegas, Nevada, 2004, pp 9–14
54. Morris K (1989) Metrics for object-oriented software development environments. Master's thesis, Sloan School of Management MIT
55. OMG Unified Modeling Language (OMG UML) (2011) Infrastructure, V2.1.1, p 158
56. Ott L, Bieman JM, Kang B, Mehra B (1995) Developing measures of class cohesion for object-oriented software. In: Proceedings of the annual Oregon workshop on software metrics (AOWSM'95), 1995

57. Ott LM, Bieman JM (1998) Program slices as an abstraction for cohesion measurement. Inf Softw Technol 40(11–12):691–699
58. Patel S, Chu WC, Baxter R (1992) A measure for composite module cohesion. In: Proceedings of the 14th international conference on software engineering, 1992, pp 38–48
59. Ponisio L, Nierstrasz O (2006) Using contextual information to assess package cohesion. Technical Report No. IAM-06-002, 2006, Institute of Applied Mathematics and Computer Sciences, University of Berne
60. Ponisio L (2006) Exploiting client usage to manage program modularity. University of Berne, Ph.D. Thesis
61. Sant'Anna C, Garcia A, Chavez C, Lucena C, von Staa A (2003) On the reuse and maintenance of aspect-oriented software: an assessment framework. In: XXIII Brazilian symposium on software engineering, Manaus, Brazil, October 2003
62. Van Solingen R (2002) The goal/question/metric approach, encyclopedia of software engineering—2 volume set, pp 578–583
63. Tagoug N (2002) Object-oriented system decomposition quality. In: Proceedings of 7th IEEE international symposium on high assurance systems engineering (HASE '02), 2002, pp 230–235
64. Telles M (2001) C# Black Book, The Coriolis Group, Nov 2001
65. The Apache Jakarta Project (2011) http://jakarta.apache.org
66. Tian J, Zelkowitz MV (1992) A formal program complexity model and its application. J Syst Softw 17:253–266
67. Wang J, Zhou Y, Wen L, Chen Y, Lu H, Xu B (2005) DMC: a more precise cohesion measure for classes. Inf Softw Technol 47(3):167–180
68. Weyuker EJ (1988) Evaluating software complexity measures. IEEE Trans Softw Eng 14(9):1357–1365
69. XGen Source Code Generator (2011) http://sourceforge.net/projects/xgen/
70. Xu B, Chen Z, Zhao J (2003) Measuring cohesion of packages in Ada95. In: Proceedings of the 2003 annual ACM SIGAda international conference on Ada: the engineering of correct and reliable software for real-time & distributed systems using Ada and related technologies, San Diego, California, USA, 2003, pp 62–67
71. Zhao J, Xu B (2004) Measuring aspect cohesion, presented at international conference on fundamental approaches to software engineering (FASE'2004)
72. Zhou Y, Wen L, Wang J, Chen Y (2003) DRC: a dependence relationships based cohesion measure for classes. In: Proceedings of the tenth Asia–Pacific software engineering conference (APSEC'03), 2003, pp 215–223
73. Zhou Y, Lu J, Lu H, Xu B (2004) A comparative study of graph theory-based class cohesion measures. Softw Eng Notes 29(2):13
74. Zhou T, Xu B, Shi L, Zhou Y, Chen L (2008) Measuring package cohesion based on context. In: Proceedings of the IEEE international workshop on semantic computing and systems, 2008, pp 127–132
75. Zuse H (1991) Software complexity: measures and methods. de Gruyter, Amsterdam