Int J Parallel Prog (2018) 46:543–564
https://doi.org/10.1007/s10766-017-0501-6

A Framework for Assessing Reusability Using Package Cohesion Measure in Aspect Oriented Systems

Puneet Jai Kaur[1] · **Sakshi Kaushal**[2] · **Arun Kumar Sangaiah**[3] · **Francesco Piccialli**[4]

Received: 5 February 2017 / Accepted: 20 March 2017 / Published online: 10 April 2017

Abstract Due to better modularization of crosscutting concerns, the Aspect oriented programming approach enhances the quality of the system as it results in less complex and more readable implementation of the system. As the software applications grow in size and complexity, they require some kind of high level organization. For high level organization of software system, packages are required. A lot of work has been carried out for measuring cohesion in Aspect Oriented Systems (AOS) at class level but very less research has been done for designing package level cohesion metric. Package cohesion metrics plays an important role in analyzing quality of software at package level. According to object oriented design principle, a good software design must have high cohesion with high reusability. Thus a relationship must therefore exist between cohesion and reusability. Number of attempts has been made to evaluate effect of cohesion on external attributes but at class level only. Impact of package level cohesion metrics on reusability for AOS is not yet explored. (a) To implement

✉ Arun Kumar Sangaiah
arunkumarsangaiah@gmail.com

Puneet Jai Kaur
puneet@pu.ac.in

Sakshi Kaushal
sakshi@pu.ac.in

Francesco Piccialli
francesco.piccialli@unina.it

1 Department of Information Technology, U.I.E.T., Panjab University, Chandigarh, India

2 Department of Computer Science and Engineering, U.I.E.T., Panjab University, Chandigarh, India

3 School of Computing Science and Engineering, VIT University, Vellore, Tamil Nadu, India

4 Department of Mathematics and Applications, University of Naples "Federico II", Naples, Italy

the proposed package cohesion measure, PCohA, on AspectJ sample packages, (b) to theoretically validate the proposed measure and (c) to find the impact of package cohesion on measuring reusability for AOS. Theoretical validation has been done by proving its validity on four theorems given by Briand et al. For finding the impact of proposed measure on external attributes, correlation has been found between package cohesion, PCohA, and external attribute—reusability. After theoretical validation, it has been proved that the proposed measure is suitable for measuring cohesion at package level. Correlation between package cohesion metric (PCohA) and reusability is calculated by using Karl Pearson Product Moment correlation. The computed values show a strong positive relation between PCohA and Reusability. The proposed package cohesion measure is found to be a useful indicator of external quality factors such as reusability. The proposed metric is also established as a better predictor of code reusability than the existing cohesion measures. The work discussed in this paper can be used for designing high quality software by developing new package level metrics for other quality attributes such as maintainability, changeability etc. as a future work.

Keywords Aspect Oriented System · Quality metrics · Package cohesion · Reusability · AspectJ

1 Introduction

Aspect Oriented Software Development (AOSD) approaches are attracting lot of attention over different domains [1] like object oriented systems, XML, etc. AOSD, as a modern programming paradigm [2] aimed at improving modularity by introducing aspects and overcomes the difficulties of Object Oriented Systems (OOS). It focuses on identification and representation of crosscutting concerns and their modularization into different working units. Some disadvantages of OOS like crosscutting concerns and separation of concerns are supported by Aspect Oriented Programming (AOP). The key difference between Object oriented programming (OOP) and AOP is that the focus of OOP is to divide programming task into objects, which encapsulate data and method, while AOP focuses on dividing the program in crosscutting concerns. These crosscutting concerns are pieces of logic that have to be applied at many places in the program, but don't have anything to do with the logic. AOSD improves separation of concerns shown by object oriented programming and support mechanisms to deal with crosscutting concerns [3].

There are many languages that belongs to Aspect oriented family such as—AspectJ, AspectXML, AspectC++, CaserJ and HyperJ. Among these AspectJ (extension of JAVA) is the most popularly used aspect oriented programming language [4]. Even though there are other AO techniques which are able to signify the concept of AOP like, Meta programming etc., AspectJ has been the most adopted technology [5]. It can be implemented with the help of seven components—Concerns, Separation of Concerns, Crosscutting Concerns, Aspect, Join points, Point cuts, Advice and Weavers [6]. Aspect-oriented system designs are decomposed into classes and aspects; aspects modularize crosscutting concerns and classes modularize non-crosscutting concerns [7]. Recently some work has been carried out for finding the pitfalls while programming AOS [8]. Their results shows that programmers in OOP stick using OO constructs they are familiar with, while avoiding AspectJ constructs and thus making mistakes. Hence measurement of quality attributes in AOP becomes necessary.

As software industry is growing day by day, the applications developed also grow in size and complexity. To accommodate the increase in size classes are not adequate to be used as the development unit for large applications. This necessitates the use of some high level of organization such as packages. The design objective of a package is to increase the quality of large software systems. Packages are highly cohesive groups that contain only related components [9]. For package to exhibit good quality software, maximum cohesion and minimum coupling among the elements is required. Cohesion is considered as the most important criteria for measuring quality of software as it is the value of similarity between different attributes of the component. An aspect in AOP is encapsulated with its state (attributes) and associated modules (operations) such as advice, introduction, point cuts and methods. The cohesion of aspect is therefore about similarity of the aspect attributes and modules. Highly cohesive components tend to have high maintainability and reusability [10]. A package in AOP is the high level abstraction of aspect. Thus a package in AOP consists of aspects or classes or interfaces which are related to one another in one or more ways.

For a software system to be good in quality and easily maintainable, it must reuse the existing packages or organize classes into packages [11]. And for better reusability, packages should contain only interconnected components with maximum cohesion and minimum coupling [12]. When packages with low cohesion are given to the software developers for reusability, they have to bring in components from other packages to provide full functionality. This will result in increased overall effort. Therefore, to overcome this problem, packages must be designed with maximum cohesion. In this case, it can be stated that the package level cohesion plays a significant role in framing software packaging and can help in analyzing external quality attributes like reusability [13]. Till date only few quantitative works have been conducted for measuring cohesion in Aspect Oriented software (AOS) and that too at aspect level only. Package level metrics can help software developers to develop high quality software. In our previous work [14] we have proposed the Package level cohesion metric for AOS. In this context, this paper, presents an extension of proposed metric as a framework for assessing reusability in AOS.

The paper is organized as follows: In Sect. 2, review of literature has been discussed. Section 3 gives the proposed package cohesion Metric (PCohA), its validation and implementation. Section 4 discusses the assessment of reusability by using the proposed metric. Section 4 is followed by conclusion and future work.

2 Literature Review

This section presents the research work carried out in the field of quality measurement for AOS. Since focus of this study is to find the cohesion at package level for AOS, metrics at package level for Object Oriented Systems have also been explored through the literature survey. Significance of any internal quality measure like cohesion can only be validated if it can help in assessing some external quality attribute like reusability etc. In this regard, we are also presenting some review work on assessing external quality attributes in AOS.

2.1 Aspect Oriented Quality Metrics

Metrics for measuring various attributes of quality like cohesion, coupling etc. have been defined at class level. Zhao [15] had proposed a framework for measuring aspect cohesion, based on the analysis of dependencies. This framework is based on the dependency graph, and it analyzes the degree of coherence between aspects attributes and modules. This approach focused on the features of the aspect itself, and does not consider the application context in which the aspect is placed. A metric suite was also designed by Sant'Anna et al. [16] on the attributes of software design and implementation. This suite includes metrics for separation of Concerns, Coupling, Cohesion and size. The proposed cohesion metric was named as *Lack of Cohesion in Operations* (*LCOO*). It measures the lack of cohesion of a component. LCOO measures the amount of method/advice pairs that do not access the same instance variable. This metric extends the CK metric Lack of Cohesion in modules (*LCOM*). The authors also framed a metrics suite for concern-driven architecture. The metric suite measures various quality attributes like cohesion, coupling and complexity for Aspect oriented and non-AOS. For cohesion, *Lack of Concern-based Cohesion* (*LCC*) metric was proposed that counts the number of concerns addressed by the assessed component.

Another framework was proposed by Ceccato and Tonella [17] for measuring Aspect oriented software development derived from C&K metric suite for object oriented systems. They had proposed cohesion metric, *Lack of Cohesion in Operations (LCO),* along with other metrics for measuring coupling. LCO was defined as the pair of operations working on different class fields minus pairs of operations working on common fields. Gelinas et al. [18] framed a metric from dependency analysis to measure *cohesion* of aspect. The metric is known as ACoh and was designed on Module-Data Connection Criterion and Module-Module Connection Criterion. A Cohesion metric for generic/unified AOS was also proposed. Here, generic means applicable to most of the AOP languages. Their framework has been defined for Java, AspectJ and CaesarJ languages. In order to represent elements of different programming languages as common and unambiguous names, they defined new terminology for AOS. They identified six different types of connections that cause cohesion. Using these connections and framework criteria, they derived cohesion metric and defined *Unified Aspect Cohesion* as UACoh. They had also correlated cohesion metric values with changeability metric and reached at the conclusion that cohesion metric cannot be for assessing changeability of the AOS. Elish et al. [19], has analyzed various function level metrics for AOS for stability assessment. An empirical validation has been carried out by Piveta et al. [20] for AOS to identify the gaps in the existing software by using the six commonly used metrics. Balani and Singh [21], has proposed SQA metrics for evaluating aspect Oriented Programs. A comparison of two paradigms i.e. Object Oriented and Aspect oriented software systems has been done by Kaur and Kaur [22] and conclusion was made that AOS are better than Object Oriented systems in certain parameters like coupling and cohesion for enhancing software design and reusability. Impact of Aspect oriented programming on Crosscutting concerns has also been explored using Breshman technique by Hans [23]. The analysis showed that modifying the functionality in any way can effect cohesion.

2.2 Package Level Cohesion Metrics

Packages are reusable components for highly organized software systems. For developing high quality AOS package level metrics are needed to be explored. Since focus of this study is to find the cohesion at package level for AOS, metrics at package level for Object Oriented Systems have been taken into consideration to reach our goal.

Measurement of cohesion at package level is very useful in software packaging as it helps in analyzing the maintainability and reusability of software. Very few metrics are available in literature to measure quality characteristics at package level. Eder et al. [24] has developed a framework to give complete criterion for cohesion at method and class level in object oriented systems. A metric suite was also proposed by Hitz and Montazeri [25] for measuring coupling in Object Oriented systems. In this suite some improvements were made on LCOM metric for measuring cohesion.

Another framework was designed by Briand et al. [26] that helps in comparing, evaluating, and defining cohesion metrics in Object Oriented systems. A new framework was proposed by Gupta and Chabbra [9] for measuring the cohesion of package in Object oriented systems. The metric depends on the relationships between package elements i.e. classes, interfaces or sub packages. During the literature review studies on package cohesion measures for Object oriented systems are found but no such study has been done for AOS. Table 1 summarizes various package level cohesion metrics that exists in literature for OO systems.

2.3 Assessment of Reusability as an External Quality Attribute

From the literature survey, it is found that most of the researchers have considered quality models for assessing external attributes. The quality model AOSQUAMO given by Kumar et al. [37], has given the significance of Reusability in measuring quality of the AOS. AOSQUAMO is the extension of the ISO/IEC 9126 (2001) software quality model. Author had added four new sub-characteristics—modularity, code-reusability, complexity, and reusability under the main characteristics of maintainability, efficiency, usability, and functionality respectively. They redefined existing characteristics of ISO/IEC 9126 in terms of AOS and highlighted the importance of the new added characteristics in AOS. The authors stated that the AOS has three categories of reusability: functional reusability, code reusability and aspect code reusability. Using their quality measurement approach, two or more projects of AOS can be compared in terms of quality.

Kumar [38] extended the AOSQUAMO model and proposed a new quality model namely AOSQ model for AOS. Author has integrated evolvability as characteristic under extensibility, sustainability, design, stability and configurability as sub characteristics into AOSQUAMO model.

Table 1 Review of package cohesion metrics for object oriented systems

Year	Author/s	Cohesion measurement
1999	Doval et al. [27]	$Cohesion = \frac{Number\ of\ intra\text{-}edge\ dependencies}{maximum\ number\ of\ possible\ dependencies}$ between the components (classes) of cluster
2000	Vernazza et al. [28]	Cohesion is calculated as the coupling of a class in a module normalized with the total coupling among the classes
2004	Khan [29]	Cohesion is measured as the total coupling amongst the classes of a package
2005	Seng et al. [30]	Cohesion is given as the number of connections between the classes in subsystem divided by the square of the number of classes in the subsystem
2005	Hussain [31]	Cohesion is calculated as the number of methods in class communicating with the methods of other classes within the same package
2009	Gui and Scott [32]	Proposed the metric that gives the degree of cohesion and transitive cohesion of methods
2009	Abdeen et al. [33]	Cohesion is described as the number of internal relations within the package
2012	Gupta et al. [9]	Cohesion is defined as number of relations between pairs of the package elements, divided by the total relations between them
2012	Singh and Bhattacherjee [34,35]	Proposed Average package level metrics by calculating the average of class level metrics
2014	Albattah and Melton [36]	Introduced package cohesion classification scheme and defined metric for each type of cohesion

Sant' Anna et al. [16], had proposed a framework for evaluating reusability and maintainability based on metric suite and quality model to assess aspect oriented software in terms of quality. Arora et al. [39], in their work has established correlation between various quality characteristics and internal quality attributes. The relation of reusability with internal quality attributes is well literated by Choudhary and Chatterjee [40] in their research work. According to authors, reusability can be assessed with help of complexity, cohesion, size and coupling metrics. They also concluded that if internal characteristics are high, then reusability is also high. It means cohesion should have positive relation with reusability. Nerurkar et al. [41], have also defined assessment model for reusability using fuzzy logic. In another research by Vinobha et al. [42], reusability is assessed with the help of inheritance metrics. They had proposed AO Reusability Evaluation Model to infer on the effect of applying inheritance in AOSD. Another model for assessing reusability in AOS is given by Singh et al. [4]. In this model the authors have given the dependency among external quality characteristics with internal quality attributes and the metrics for measuring these internal attributes. The dependency of reusability on other quality attributes is also given by Dhole and Nirmal [43]. According to them, reusability is dependent on portability, adaptability, maintainability and understandability. A framework for assessing maintainability has been proposed by Ananthi and Roby [44]. The framework suggested to use the static metrics for calculating maintainability. Mallikarjuna Reddy et al. [45] in their work has given the importance of maintenance and understandability in AOS. They had presented various metrics for calculating these attributes.

From the detailed review of literature, it is found that no work has been done in the field of measuring cohesion at package level in AOS. Although lot of work is being carried on assessing various external quality attributes, but no contribution is found in regard to using package cohesion to assess quality attributes. To overcome this research gap, we have proposed the package cohesion metric (PCohA) in our previous work [14]. In this paper, we are defining a framework for using the proposed package cohesion metric for assessing reusability in AOS.

3 Package Cohesion Metric

This section gives the theoretical framework for defining the package cohesion metric. It includes the basic definitions and relationships associated with package that helped in building the framework as discussed in following sections.

A package can be defined as the group of elements and relations between the elements. The elements of a package for AOP includes aspects, classes, interfaces, and sub-package. The presence of a sub-package in a package creates a leveled structure of the package. At level i, package is defined as given in Eq. (1):

$$p_i = [E_i, R_i] \tag{1}$$

where E_i, represents the elements of package i and R_i represents the set of relations on E_i.

3.1 Types of Relationships

A package in AOP consists of aspects, classes, interfaces or sub-packages. If two elements x and y of a package are related to each other, their relation is shown as $r(x, y) = 1$. It means element x is related to element y. The relations are asymmetric in nature, $i.e.\, x \rightarrow y$ does not means $y \rightarrow x$. As there are different types of elements in a package, there are different types of relations between them. Aspects and classes are similar in their functioning and structure where aspects modularize crosscutting concerns and classes modularize non-crosscutting concerns. Interfaces are again handled similarly as classes. Thus different relations that can happen are: *Aspect–Aspect*, *Class–Class*, *Aspect–Class*, *Class–Aspect*, *Aspect–Subpackage*, *Subpackage–Aspect*, *Subpackage–Class*, *Class–Subpackage*, *Subpackage–Subpackage.*

3.2 Cohesion Measurement at Package Level

Package cohesion is measured as an intra package dependency amongst its elements. Intra package dependency considers all elements used directly or indirectly by the package. Inspired by some approaches for defining class cohesion measures [18] package cohesion metric PCohA can be defined as the relatedness between the package

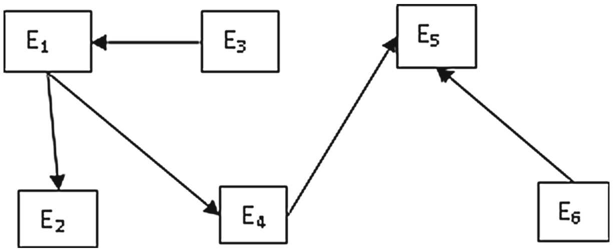


Fig. 1 Example of connections between elements of package

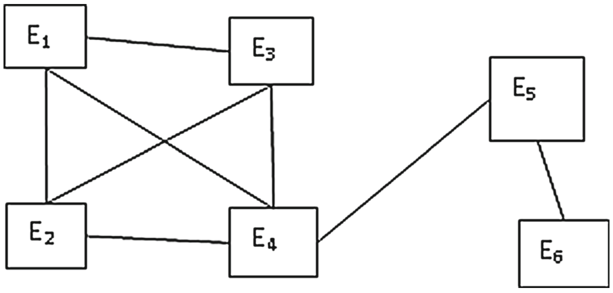


Fig. 2 Undirected connection graph

members. This relatedness is measured as the total number of connections among the package components—both direct and indirect. Figure 1 shows the relationship diagram between elements of a package.

Suppose E_1, E_2, E_3, E_4, E_5 and E_6 are the components of a package, which can be an aspect, an interface or a class and there are some dependencies between them. E_1 depends on E_2, E_4 and E_5. In other words, E_1 depends directly on E_2 and E_4 whereas it depends on E_5 indirectly through E_4. Similarly, E_3 indirectly depends on E_2 and E_4 through direct connection with E_1. If we consider the graph with all connections (direct and indirect) for Fig. 1, we get undirected graph shown in Fig. 2.

Based on this dependency criteria, we have proposed cohesion metric at package level as the total number of dependent edges in the undirected connection graph (NCE) divided by maximum number of connections between elements (MCE) [14], as shown below in Eq. (2)

$$PCohA = \frac{NCE}{MCE} \in (0, 1) \tag{2}$$

where, NCE is the count of connected edges and MCE is calculated as $N \times (N - 1)$, if N is the number of elements in the package.

Figure 3 illustrates the various steps to measure the proposed metric PCohA. Following these steps, value of PCohA is calculated for an example shown in Fig. 2. In Fig. 2, there are 6 elements (aspects/classes/interfaces). Connected edge gives the relation among elements. There are 8 connected edges/relations among elements of this example. Hence from Eq. (2)

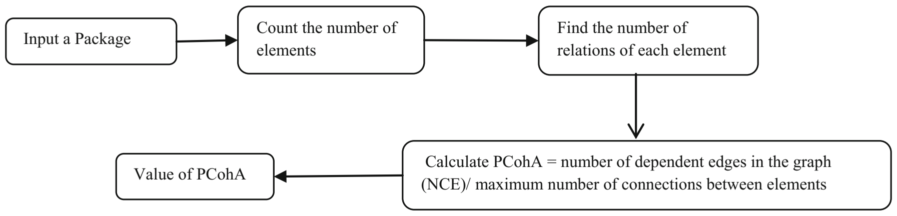

Input a Package
Count the number of elements
Find the number of relations of each element
Calculate PCohA = number of dependent edges in the graph (NCE)/ maximum number of connections between elements
Value of PCohA

Fig. 3 Demonstration of proposed cohesion metric

Table 2 Qualitative categorization of cohesion [46]

PCohA range	Cohesion category
0	Non-cohesive
$0 < \text{PCohA} \leq 0.3$	Loosely cohesive
$0.3 < \text{PCohA} < 0.7$	Averaged cohesive
$0.7 < \text{PCohA} < 1$	Strongly cohesive
1	Highly cohesive

$$\begin{aligned}\text{PCohA} &= 8/6(6-1)\\ &= 8/30\\ &= 0.26\end{aligned}$$

Since the value of PCohA is near to 0, this package is loosely cohesive [46].

A package is said to be highly cohesive if value of PCohA is 1 and non-cohesive if PCohA is 0. Table 2 represents the relation between qualitative and quantitative value for cohesion metric.

From the Table 2 description, it can be concluded that the cohesion of a package is desirable to be near to 1 and if it is near to 0, it is less cohesive

The proposed package cohesion metric, PCohA, is analysed by considering three cases as:

Worst Case When every element within a package is independent or package is empty, $i.e.$ if, N = 0, then there will be no relation and hence $PCohA = 0$.

Best Case When every class within a package depends on other classes of the same package, it will contain all the possible relations it can and hence, $PCohA = 1$.

Average Case When approximately half of the classes of a package are related with every class of package.

The proposed metric, PCohA, is further theoretically validated using four properties given by Briand et al. in 1996.

3.3 Theoretical Validation of Proposed Metric

Briand et al. [26] suggests that cohesion measurement must follow some set properties. These properties are: Non-negativeness, normalization, minimum and maximum values, monotonicity and merging property.

First two properties are self-explanatory from the given definition of PCohA in Eq. (2). The value of package cohesion measured by proposed metric will always fall between 0 and 1, i.e., the value is normalized and it will always be positive.

If package P is empty, it means there is no element in the package i.e. $N = 0$ and there will be no relation. In this case, $PCohA(P) = NULL$. If number of elements in package is N and assuming that each element of package is related to all other elements of the package, then $PCohA(P) = 1$ and if $N = 1$, i.e. package has only one element then also $PCohA(P) = 1$, i.e maximum value. It can thus be concluded that the proposed metric has minimum value 0 and maximum value 1.

Monotonicity means on adding relationships in package, cohesion must not decrease. It requires the addition of relations not elements. Let package P_1 contains relations R_1 and P_2 is the modified package with added relations. If R_2 is the number of relations in package P_2, then R_1 belongs to R_2. According to the proposed measure, the value of numerator will never decrease but can either increase or stay same. Hence, if R_1 belongs to R_2 then, $PCohA(P_1) \leq PCohA(P_2)$ i.e. cohesion will never decrease with the addition of relations.

According to merging property, cohesion value should not increase by combining unrelated packages. If P_1 and P_2 are two unconnected packages and P is the package formed after merging these two packages, then according to the proposed measure, the value for numerator in PCohA (P) will increase in less amount as compared to the value of denominator. In this case, package cohesion may decrease i.e., $max\,[PCohA\,(P_1)\,, PCohA\,(P_2)] \geq PCohA\,(P)$.

The proposed metric is theoretically validated since all the four properties are satisfied. We have implemented our metric PCohA on AspectJ projects available as AspectJ benchmarks for research use (http://abc.comlab.ox.uk/) and some projects are enclosed within the eclipse platform as AspectJ examples (https://eclipse.org/aspectj/doc/released/progguide/examples.html). Table 3 gives the details of AspectJ projects taken for research work.

3.4 Implementation of Package Cohesion Metric

Here we are implementing proposed metric PCohA on AspectJ Projects given in Table 3. UML notations have been used for counting number of connected edges [47]. AspectJ projects Bean Aspect and Subject Observer Protocol have been taken as examples for illustrating the calculation of PCohA.

3.4.1 Bean Aspect

The Bean example describes an aspect that makes Point objects into java bean with bound properties. The example contains three classes—point, BoundPoint and Demo. Point is a simple class representing points with rectangular coordinates. The BoundPoint is an aspect responsible for Point to be a bean and the Demo class is a test program that act as a property change listener for a point object that it creates and then performs simple updations on it.

Table 3 AspectJ projects

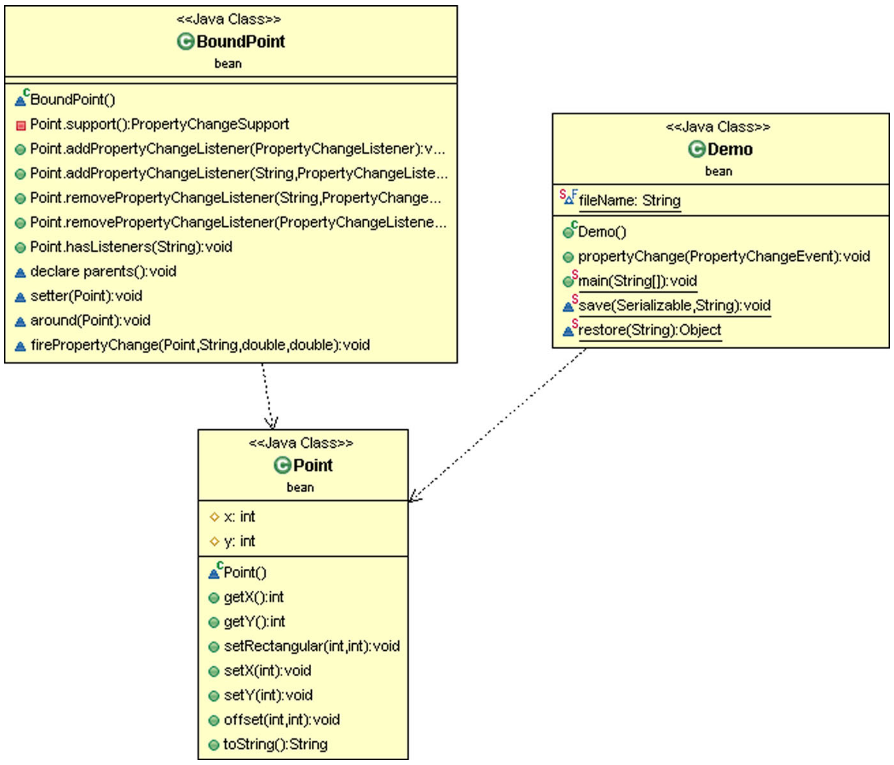
Package no.	AspectJ projects	No. of packages		Size (LOC)
P1	Bean example	1		125
P2	Introduction	1		148
P3	Observer figure	1		132
P4	Spacewar	2	Coordination	317
P5			Spacewar	1110
P6	Telecom	1		181
P7	TJP	1		50
P8	Tracing	1		84
P9	Observer	1		83
P10	Producer consumer	1		51
P11	Resource pool management	1		125
P12	SBT-ATM	2	ATM	54
P13			ATMtest	45
P14	DCM	5	DCM.certrevsim	974
P15			DCM	116
P16			DCM.handleGC	33
P17			DCM.Jsim.event	64
P18			DCM.Jsim.queue	416
P19	Figures	1		94
P20	NullCheck	3	NullCheck.Certrevsim	974
P21			NullCheck.Jsim.event	64
P22			NullCheck.Jsim.queue	416
P23	Quicksort	1		72

Figure 4 shows the UML diagram for Bean Aspect. It is clear from the diagram that BoundPoint aspect and Demo class are related to Point Class. It means the Bean example has two relations and three elements. The proposed measure of Package cohesion for Bean example is calculated as:

$$\begin{aligned} PCohA &= 2/3(3-1) \\ &= 2/6 \\ &= 0.33 \end{aligned}$$

3.4.2 The Subject Observer Protocol

The Observer design pattern defines dependency between a subject and several observers. When the subject changes its state, all observer objects will automatically be notified and updated accordingly. The basic parts of the protocol are the interfaces Subject and Observer, and the aspect Subject Observer Protocol.



<<Java Class>>
BoundPoint
bean
BoundPoint()
Point.support():PropertyChangeSupport
Point.addPropertyChangeListener(PropertyChangeListener):v...
Point.addPropertyChangeListener(String,PropertyChangeListe...
Point.removePropertyChangeListener(String,PropertyChange...
Point.removePropertyChangeListener(PropertyChangeListene...
Point.hasListeners(String):void
declare parents():void
setter(Point):void
around(Point):void
firePropertyChange(Point,String,double,double):void
<<Java Class>>
Demo
bean
fileName: String
Demo()
propertyChange(PropertyChangeEvent):void
main(String[]):void
save(Serializable,String):void
restore(String):Object
<<Java Class>>
Point
bean
x: int
y: int
Point()
getX():int
getY():int
setRectangular(int,int):void
setX(int):void
setY(int):void
offset(int,int):void
toString():String

Fig. 4 UML for Bean Aspect

The Subject interface contains methods to add, remove, and view Observer objects and the Observer interface contain methods to set and get Subject objects. The aspect SubjectObserverProtocol contains all the basic parts to execute the update method of Observer object when some state changes in a subject. It defines an abstract pointcut that is overridden on extending aspects. It also defines an advice that run after the join points of the pointcut. And it declares an inter-type field and two inter-type methods so that each Observer can hold onto its Subject. Button object makes sure that the void click () method is called whenever a button is clicked. This class knows nothing about being a Subject. ColorLabel objects are labels that support the void colorCycle() method. Again, they know nothing about being an observer. Finally, the SubjectObserverProtocolImpl implements the subject/observer protocol, with Button objects as subjects and ColorLabel objects as observers.

From Fig. 5, we can predict that Subject/Observer protocol has 8 elements and 14 relations between them. So Package Cohesion as calculated from our proposed measure is:

$$\begin{aligned} \text{PCohA} &= 14/8(8-1) \\ &= 14/56 = 0.25 \end{aligned}$$

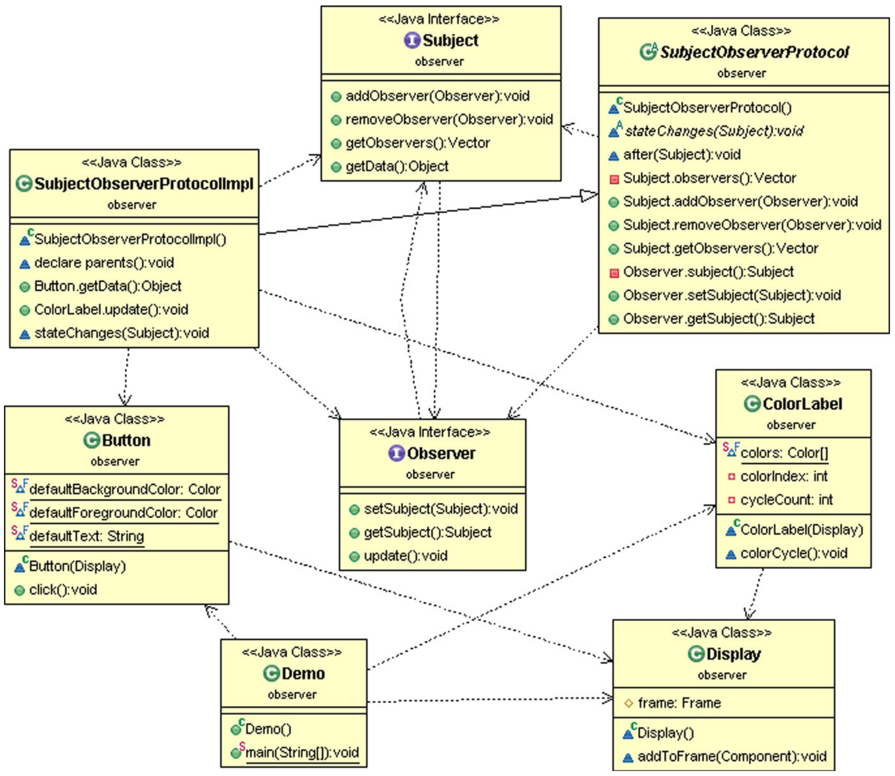


Fig. 5 UML for Subject Observer Protocol

The value of PCohA for other AspectJ projects is calculated in similar manner and is given in Table 4.

From PCohA values obtained for the examples in Table 4 and the qualitative categorization from Table 2, it is concluded that 10 packages are loosely cohesive, 11are averaged cohesive and 2 are highly cohesive.

In the next section, we are validating PCohA against external attribute i.e. reusability for proving its significance in calculating the quality of AOS.

4 Assessing Reusability Through Package Cohesion Measure

Software quality is the phenomenon for meeting system's specified requirements. As shown in Fig. 6, measure of any internal quality attribute is significant only if it can assess some external quality attribute. In concepts of software engineering, measures of internal quality attributes have no meaning if it cannot measure any external quality attribute [48].

Internal quality attributes may include the various metrics for measuring cohesion, coupling, size, separation of concerns etc. and some main external quality attributes extracted from the literature are—reusability, maintainability, understandability and

Table 4 PCohA values for AspectJ examples

Package	No. of classes	No. of interfaces	No. of relations	PCohA
P1	3	0	2	0.33
P2	4	0	3	0.25
P3	6	2	14	0.25
P4	6	3	11	0.15
P5	23	0	26	0.05
P6	7	0	16	0.38
P7	2	0	1	0.5
P8	4	0	4	0.33
P9	8	2	14	0.15
P10	2	0	1	0.5
P11	5	0	5	0.25
P12	2	0	2	1
P13	4	0	8	0.66
P14	13	0	31	0.19
P15	2	0	1	0.5
P16	0	1	0	1
P17	3	0	3	0.5
P18	11	0	26	0.23
P19	5	1	10	0.33
P20	13	0	31	0.19
P21	3	0	3	0.5
P22	11	0	26	0.23
P23	**3**	**0**	**2**	**0.33**

The values for the metric PCohA calculated for the sample examples are given in bold

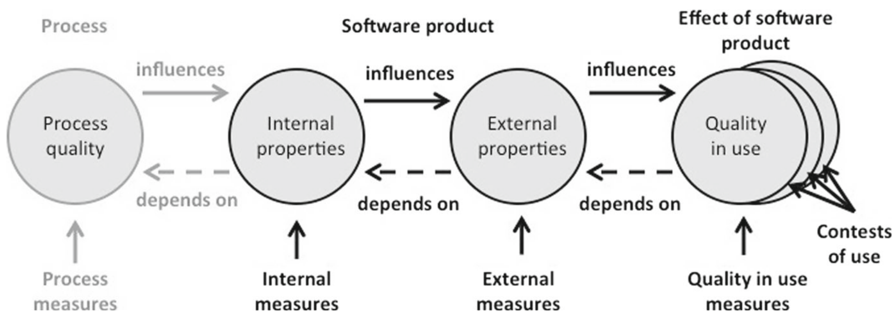


Fig. 6 Quality lifecycle (https://xbosoft.com/definition-software-quality)

testability. Here, in this paper external quality attribute—reusability, is assessed using package cohesion metric to prove its significance in AOSD.

Reusability of the software systems is becoming a very important factor due to rapid software development and increasing complexity. Software reusability is a process of

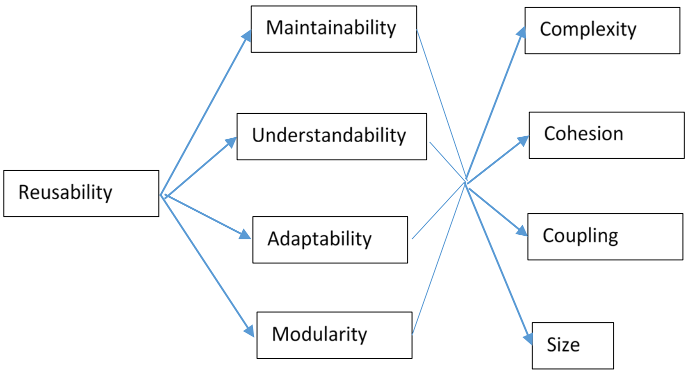


Fig. 7 Relation of reusability with external and internal quality attributes

reusing the software with very little or no modification. The software is reusable if it is easily understandable and is less complexed. In terms of AOS, reusability is defined as the capability of an AO component to be reused in other AO components. AOP overcomes the major limitation of OOP *i.e.* crosscutting concerns. In OOP, the code related to concern is scattered in multiple classes, affect reusability and maintainability. These scattered concerns are called crosscutting concerns. AOP provides solution to the problem by encapsulating code of crosscutting concern in a single module called aspect. The aspects encapsulating crosscutting concern are integrated with classes (primary classes) with the help of weaving process. Weaving injects the code of an aspect into well-defined locations called joinpoints in primary concerns (classes). This is the way of achieving reusability on AOP [41].

Software reusability improves the quality of software product by reducing development time, effort and cost. From the literature review it is found that, various models have been proposed for Reusability. All models concentrate on relating Reusability with various external quality attributes and measuring them using internal characteristics. It is concluded that Reusability is highly dependent on the four main quality attributes—maintainability, Understandability, Adaptability and Modularity. All these four external attributes can be measured with the help of internal attributes like complexity, coupling, cohesion, size etc. Figure 7 illustrates the importance and significance of cohesion in measuring reusability.

4.1 The Assessment Framework

The measure of any internal attribute, such as cohesion, is significant only if it can help in assessing some external attribute (e.g. reusability). In software development, internal quality attributes are superficial concepts and, in themselves, have no meaning. Thus, there is a requirement to develop a framework to assess internal attributes like, cohesion and size in terms of their usefulness as indicator of the external qualities like, maintainability and reusability. In fact, the aim of the assessment framework is to provide support for assessment of reusability of aspect-oriented systems based on the proposed metric. The framework components help organize the assessment process

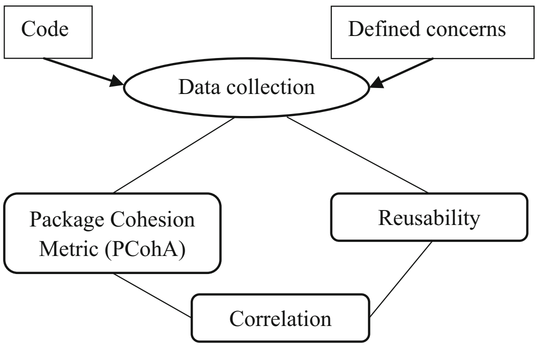


Fig. 8 The assessment framework

Table 5 Design metrics for AOS

Design property	Design metric
Coupling	Efferent coupling (CE)
Cohesion	Package cohesion metric (PCohA)
Messaging	Interface size of package (IP)
Design size	Number of classes and interfaces (ADS)

and assist in data collection and interpretation (Fig. 8). The basic components of the framework are: package level metrics and the reusability model (Fig. 7). The reusability model establishes the relationships between the external attributes, internal attributes and the metrics. The framework requires some artifacts as inputs to the measurement process. First, it requires the system code for the use of the metrics. In addition, the assessment framework requires a description of the system concerns to guide the identification of the concerns when using the metrics of separation of concerns.

In this paper, we are using our proposed design metrics [14] as given in Table 5, for calculating reusability. Reusability can be calculated as given in Eq. (3):

$$\begin{aligned}\text{Reusability} = &-0.25 \times \mathit{Coupling} + 0.25 \times \mathit{Cohesion}\\ &+ 0.5 \times \mathit{Messaging} + 0.5 \times \mathit{Design\ Size}\end{aligned} \tag{3}$$

Table 6 gives the value of reusability for each package mentioned in Table 3, calculated by applying metrics defined in Eq. 3.

4.2 Correlation Between PCohA and Reusability

For finding the correlation between Reusability and PCohA, rating is given to values of Reusability. Values are rated on a numerical scale from 1 (high) to 10 (low). The package with high value of reusability is easy to be reused and hence rated as 1 as presented in Table 7.

Figure 9 shows the relationship between values of package cohesion (PCohA) and reusability for each package.

Table 6 Reusability of AspectJ projects

Package	CE	PIS	DSP	PCohA	Reusability
P1	0	1	3	0.33	2.0825
P2	0	4	4	0.25	4.0625
P3	0	1	8	0.25	4.5625
P4	0	4	9	0.15	6.5375
P5	2	1	23	0.05	11.5125
P6	0	13	7	0.38	10.095
P7	1	1	2	0.5	1.375
P8	0	4	4	0.33	4.0825
P9	0	4	10	0.15	7.0375
P10	0	2	2	0.5	2.125
P11	0	3	5	0.25	4.0625
P12	0	2	2	1	2.25
P13	1	1	4	0.66	2.415
P14	3	13	13	0.19	12.2975
P15	0	3	2	0.5	2.625
P16	1	1	1	1	1
P17	1	3	3	0.5	2.875
P18	0	11	11	0.23	11.0575
P19	0	0	6	0.33	3.0825
P20	3	13	13	0.19	12.2975
P21	1	3	3	0.5	2.875
P22	0	11	11	0.23	11.0575
P23	0	2	3	0.33	2.5825

It is clear from the Fig. 9 that the PCohA has positive relation with Reusability. Reusability is high for the package with high cohesion. Highly cohesive package P7, P10, P12, P13, P16, P17 and P21 has ratings between 8 and 10, i.e., high reusability. Similarly, less cohesive packages like P5 and P9 has ratings between 2 and 4, i.e., less reusable.

Correlation between package cohesion metric (PCohA) and reusability is calculated by using Karl Pearson Product Moment correlation from the values in Table 7. Correlation values are as shown in Table 8.

The computed values show a strong positive relation between PCohA and reusability. Thus it can be concluded that level of PCohA can determine the level of reusability in AOS.

4.3 Comparison with Existing Measures

From the literature review, it is found that no work has been done on designing package cohesion metric for AOS. Commonly package cohesion can be calculated by taking

Table 7 Ratings for reusability

Package	Reusability	Rating	PCohA
P1	2.0825	8	0.33
P2	4.0625	6	0.25
P3	4.5625	6	0.25
P4	6.5375	5	0.15
P5	11.5125	2	0.05
P6	10.095	3	0.38
P7	1.375	9	0.5
P8	4.0825	6	0.33
P9	7.0375	4	0.15
P10	2.125	8	0.5
P11	4.0625	6	0.25
P12	2.25	8	1
P13	2.415	8	0.66
P14	12.2975	1	0.19
P15	2.625	7	0.5
P16	1	9	1
P17	2.875	7	0.5
P18	11.0575	2	0.23
P19	3.0825	7	0.33
P20	12.2975	1	0.19
P21	2.875	7	0.5
P22	11.0575	2	0.23
P23	2.5825	8	0.33

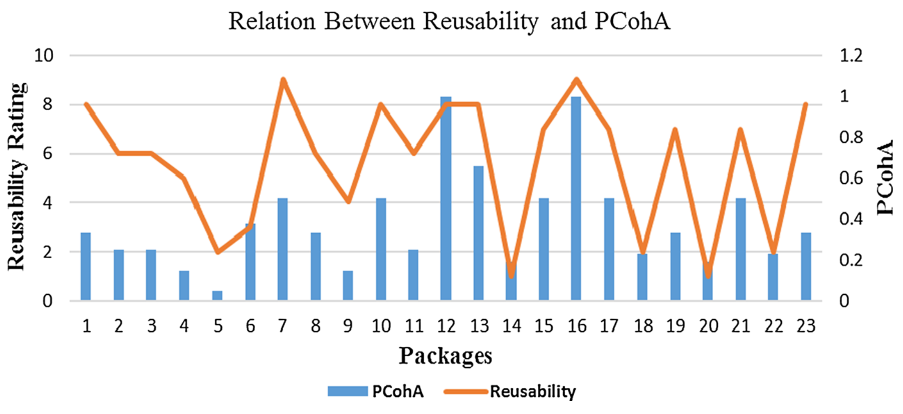


Fig. 9 Relation between Reusability and PCohA

the average of cohesion measure, *Lack of Cohesion in Objects* (*LCOO*), given by Sant'Anna et al. [16] for each package in the project. We are comparing LCOO with our proposed metric. Table 9 gives the value for LCOO for selected packages.

Table 8 Correlation values

Attributes	Correlation value	Level of significance
PCohA and reusability	0.67	0.05

Table 9 9 values of LCOO

Package	LCOO	Package	LCOO
P1	0.17	P13	0
P2	0.16	P14	0.23
P3	0	P15	0
P4	0.4	P16	0
P5	0.27	P17	0.16
P6	0.28	P18	0.25
P7	0	P19	0.2
P8	0.06	P20	0.23
P9	0	P21	0.16
P10	0	P22	0.25
P11	0	P23	0.13
P12	0		

Table 10 Correlation values for PCohA and LCOO

Attributes	PCohA	LCOO
Reusability	0.67	−0.64

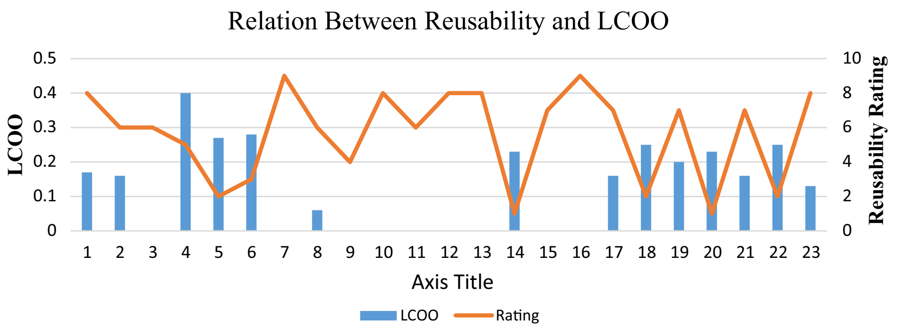


Fig. 10 Relation between Reusability and LCOO

Figure 10 shows the relation between LCOO values and Reusability ratings obtained for selected packages. The correlation values at significance level 0.05 are shown in Table 10.

Table 10 shows that the proposed measure PCohA has produced high value of correlation with reusability, i.e. 0.67. LCOO has also shown good correlation, though it is negative, i.e., −0.64, as this metric measures lack of cohesion. But our proposed approach is much suitable as it is less complex to calculate. Using LCOO to measure package cohesion, it has to be repeatedly calculated for all elements of package which is very cumbersome. Whereas using PCohA for same purpose requires calculation for one time only, reducing time and effort. Hence, it can be concluded that PCohA is much suitable for measuring the reusability in AOS.

4.4 Threats to Validity

As in other empirical studies, there are threats to validity of this study also. The use of one AOP language i.e. AspectJ can be a threat in our study. However, it is fairly known that AspectJ systems are being used in majority of AOS studies. Even though there are other AO techniques which are able to signify the concept of AOP like, Meta programming etc., AspectJ has been the most adopted technology [5]. Another threat regards the number and size of the target applications chosen for proving validation of our metric. Regarding the size of our selected examples, small applications are taken, as it is a beginning in the category of defining package level metrics. To reduce this threat, in future large applications can be used and also more number of package can be used to further analyze the metric. The major threat is the use of Reusability as an indicator of external software quality. The external attributes defining quality of a software are also dependent on maintainability, understandability etc. To reduce this limitation, other measures like maintainability and understandability will be used to validate the metric.

5 Conclusion and Future Work

In this paper, we have made an attempt to measure cohesion at package level in AOS. We have proposed a new metric, PCohA, for measuring package level cohesion in AOS. The proposed metric PCohA is based on formal definitions and relations amongst the elements of package. The proposed metric is theoretically validated as it satisfies the four properties given by Briand et al. [26]. The metric is implemented on 23 AspectJ packages available as an open source in AspectJ repository and some embedded with Eclipse platform. To prove the usefulness of proposed measure a framework has been defined to empirically validated it on external quality attribute—reusability. The empirical evaluation has clearly shown that the proposed metric has a positive effect on the measurement of reusability and it can be used as an assessment tool for measuring reusability of AOS. The work discussed in this paper can be used for designing software with high quality attributes. The results of this study can also be used for helping in the development and improvement of new AO packages.

This proposed measure can be used in future to develop new package level metrics for other software quality attributes such as understandability, maintainability, changeability etc. New replications of the experiment can be performed using large datasets and other AO languages.

References

1. Ali, M.S., Babar, M.A., Che, L., Stol, K.J.: A systematic review of comparative evidence of aspect oriented programming. Inf. Softw. Technol. **52**, 871–887 (2010)
2. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Longtier, J.M., Irwin, J.: Aspect-oriented programming. In: Proceedings of the European Conference on Object Oriented Programming (ECOOP), Springer, LNCS 1241 (1997)
3. Cacho, N., Santanna, C., Figueiredo, E., Dantas, F., Garcia, A., Batista, T.: Blending design patterns with aspects: a quantitative study. J. Syst. Softw. **98**, 117–139 (2014)
4. Singh, P.K., Sangwan, O.P., Singh, A.P.: A quantitative evaluation of reusability for aspect oriented software using multi-criteria decision making approach. World Appl. Sci. J. **30**(12), 1966–1976 (2014)
5. Rashid, A., Cottenier, T., Greenwood, P., Chitchyan, R., Meunier, R., Coelho, R., Sudholt, M., Joosen, W.: Aspect oriented programming in practice: tales from AOSE-Europe. Computer **42**(2), 19–26 (2010)
6. Brichau, J., D'Hondt, T.: An introduction to Aspect Oriented Software Development. AOSD Europe (2005)
7. Fabry, J., Roover, C., Noguera, C., Zschaler, S., Rashid, A., Jonckers, V.: AspectJ code analysis and verification with GASR. J. Syst. Softw. **117**, 528–544 (2016)
8. Santos, A., Alves, P., Figueiredo, E., Ferrari, F.: Avoiding code pitfalls in aspect-oriented programming. Sci. Comput. Program. **119**, 31–50 (2016)
9. Gupta, V., Chhabra, J.K.: Package level cohesion measurement in object oriented software. J. Braz. Comput. Soc. **18**, 251–266 (2012)
10. Ebad, S., Ahmed, M.: An evaluation framework for package level cohesion metrics. In: International Conference on Future Information Technology, vol. 13, pp. 239–243. IACSIT Press, Singapore (2011)
11. Almugrin, S., Albattah, W., Melton, A.: Using indirect coupling metrics to predict package maintainability and testability. J. Syst. Softw. **121**, 298–310 (2016)
12. Tahir, A., Ahmad, R.: An AOP based approach for collecting software maintainability dynamic metrics. In: Second international Conference on Computer Research and Development. IEEE, pp. 168–172 (2010)
13. Tahir, A., Ahmad, R., Kasirun, Z.: Maintainability dynamic metrics data collection based on aspect oriented technology. Malays. J. Comput. Sci. **23**(3), 177–194 (2010)
14. Kaur, P.J., Kaushal, S.: Package level metrics for reusability in AOS. In: International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (A BLAZE), pp. 364–368. IEEE, Amity University, Noida, 25–27 Feb 2015
15. Zhao, J.: Towards a metric suite for aspect oriented software. Technical report, SE 136-25, Information Processing Society of Japan (IPSJ) (2002)
16. Sant'Anna, C., Garcia, A., Chavez, C., Lucena, C., Staa, A.: On the reuse and maintenance of aspect oriented software: an assessment framework. In: 17th Brazilian Symposium on Software Engineering (2002)
17. Ceccato, M., Tonella, P.: Measuring the effects of software aspectization. In: Proceedings of First Workshop on Aspect Reverse Engineering, WARE (2004)
18. Gelinas, J.F., Badri, M., Badri, L.: A cohesion measure for aspects. J. Object Technol. **5**(7), 97–114 (2006)
19. Elish, M.O., Al-Khiaty, M., Alshayeb, M.: Investigation of aspect-oriented metrics for stability assessment: a case study. JSW **6**(12), 2508–2514 (2011)
20. Piveta, E.K., Moreira, A., Pimenta, M.S., Araujo, J., Guerreiro, P.: An Empirical Study of Aspect Oriented Metrics, Science of Computer Programming, vol. 78. Elsevier, Amsterdam (2012)
21. Balani, L., Singh, A.: Software quality metrics for aspect oriented programming. Int. J. Eng. Res. Technol. **8**(1), 1–6 (2015)
22. Kaur, M., Kaur, R.: Improving the design of Cohesion and coupling metrics for aspect oriented software development. Int. J. Comput. Sci. Mob. Comput. **4**(5), 99–106 (2015)
23. Hans, A.: Impact of aspect oriented programming on cross cutting metrics using Breshman technique for homogeneity. Int. J. Adv. Res. Electron. Commun. Eng. **5**(8), 2172–2178 (2016)
24. Eder, J., Kappel, G., Schrefl, M.: Coupling and cohesion in object oriented system. Technical report, University of Klagenfurt, Austria (1994)
25. Hitz, M., Montazeri, B.: Measuring coupling and cohesion in OO systems. In: Proceedings of International Symposium on Applied Corporate Computing, Monterrey, Mexico (1995)

26. Briand, L.C., Morasca, S., Basili, V.R.: Property based software engineering measurement. IEEE Trans. Softw. Eng. **22910**, 68–86 (1996)
27. Doval, D., Mancoridis, S., Mitchell, B.S.: Automatic clustering of software systems using genetic algorithm. In: STEP'99, pp. 73–41. IEEE Computer Society (1999)
28. Vernazza, T., Granatella, G., Succi, G., Benedicenti, L., Mintchev, M.: Defining metrics for software components. In: The World Multiconference on Systemics, Cybernetics and Informatics, Florida (2000)
29. Khan, S.: Design level coupling metrics for UML models. MS Thesis, KFUPM, Saudi Arabia (2004)
30. Seng, O., Bauer, M., Biehl, M., Pache, G.: Search-based improvement of subsystem decompositions. In: GECCO'05, pp. 1045–1051 (2005)
31. Hussain, S.: Package cohesion metric for OO systems. MS Thesis, KFUPM, Saudi Arabia (2005)
32. Gui, G., Scott, P.D.: Coupling and cohesion metric for evaluation of software component reusability. In: International Conference on Young Computer Scientists, pp. 1181–1186. IEEE (2008)
33. Abdeen, H., Ducasse, S., Sahraoiy, H., Alloui, I.: Automatic package coupling and cycle minimization. In: WCRE'09, CNF, pp. 103–122. IEEE (2009)
34. Singh, V., Bhattacherjee, V.: Evaluation and application of package level metrics in assessing software quality. Int. J. Comput. Appl. **58**(21), 38–46 (2012)
35. Singh, V., Bhattacherjee, V.: Assessing package reusability in object oriented design. Int. J. Softw. Eng. Appl. **8**(4), 75–84 (2014)
36. Albattah, W., Melton, A.: Package cohesion classification. In: 5th IEEE International Conference of Software Engineering and Service Science (ICSESS), Beijing, pp. 1–8 (2014)
37. Kumar, A., Kumar, R., Grover, P.S.: Towards a unified framework for cohesion measurement in AOS. In: 19th Australian Conference on Software Engineering, pp. 57–65. IEEE (2008)
38. Kumar, P.: Aspect oriented software quality model: the AOSQ model. Adv. Comput. **3**(2), 105–118 (2012)
39. Arora, K., Singhal, A., Kumar, A.: A study of cohesion metrics for Aspect Oriented System. Int. J. Eng. Sci. Adv. Tech. 2(2), 332–337 (2012)
40. Choudhary, R., Chatterjee, R.: Reusability in AOSD—the aptness, assessment and analysis. In: International Conference on Reliability, Optimization and Information Technology (ICROIT 2014). IEEE, pp. 34–39, 6–8 Feb 2014
41. Nerurkar, N.W., Kumar, A., Shrivastava, P.: Assessment of reusability in AOS using fuzzy logic. ACM SIGSOFT Softw. Eng. Notes **35**(5), 1–5 (2010)
42. Vinobha, A., Velan, S., Babu, C.: IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 1715–1722 (2014)
43. Dhole, A., Nirmal, N.: An approach for calculation of reusability metrics of object oriented program. Int. J. Eng. Res. Technol. **2**(6), 2644–2647 (2013)
44. Ananthi, S., Roby, J.: A theoretical framework for the maintainability model of AOS. In: International conference on Soft Computing and Software Engineering (SCSE), Procedia Computer Science, vol. 62, pp. 505–512 (2015)
45. Mallikarjuna Reddy, G., Anil Babu, N., Arun Kumar, R., Deshmukh, G.: Maintenance and understandability of aspect oriented programming. Int. J. Comput. Trends Technol. **36**(2), 77–80 (2016)
46. Tripathi, A., Vardhan, M., Kushwaha, D.S.: Package level cohesion and its application. In: Proceedings of International Conference on Advances in Communication, Network and Computing, CNC, pp. 437–446. Elsevier (2014)
47. Garg, S., Kahlon, K.S., Bansal, P.K.: How to measure coupling in AOP from UML diagram. Int. J. Comput. Sci. Telecommun. **2**(8), 52–57 (2011)
48. Briand, L., Emam, K.E., Morasca, S.: Theoretical and empirical validation of software product measures. Technical report ISERN-95-03. Fraunhofer Institute of Experimental Software Engineering, Germany (1995)