A Critical Review of Cohesion Measures and Measurement Frameworks in Aspect-Oriented Systems

Avadhesh Kumar
Amity Institute of Information Technology
Amity University, Sec-125, Noida, India, avadheshkumar@aiit.amity.edu

Rajesh Kumar
School of Mathematics & Computer Applications
Thapar University, Patiala, Punjab, India, rakumar@tiet.ac.in

P.S. Grover
Guru Tegh Bahadur Institute of Technology
GGS Indraprastha University, Delhi, India, groverps@hotmail.com

ABSTRACT
Cohesion is considered as one of the most important software quality attributes. Cohesion refers to the degree of relatedness between members of a software component. Cohesion is one of the few internal software attributes that has been both theoretically and empirically shown to have large impact on external software quality characteristics like maintainability, modularity, reusability, testability and usability. Aspect-Oriented Programming (AOP) is an emerging technique that provides a means to cleanly encapsulate and implement aspects that crosscut other modules. Aspects are linguistic entities aimed at describing a crosscutting concern. However, despite an interesting body of work for measuring cohesion in Aspect-Oriented (AO) Systems, there is poor understanding of cohesion in the contest of AOP. Some researchers and practitioners have proposed cohesion measures, cohesion measurement/assessment frameworks and metrics for AOP. But most of the framework and metrics are for AspectJ programming language. They have defined cohesion in their frameworks and metrics in terms of AspectJ programming language. In this paper we have critically reviewed and compared exiting cohesion measures on the basis of cohesion measurement criterions. We found, most of the existing measures and measurement frameworks are for the family of AspectJ (AspectJ-like) programming languages, there is no any existing generic cohesion framework that takes into account other families of AOP languages such as CaesarJ and Hyper/J. So there is need to define a generic/unified cohesion measurement framework. The unified framework could contribute in better understanding of cohesion in AOP. This will be helpful in (i) comparing measures and their potential use, (ii) integrating different existing measures which examine the same concept in different ways, and (iii) defining new cohesion metrics, which in turn permits the analysis and comparison of Java, AspectJ and CaesarJ implementations.

Keywords: cohesion, aspect-oriented programming, software metric.

1. INTRODUCTION
Software quality refers to the conformance of the product to explicitly state functional and performance requirements, documented development standards, and implicit characteristics. Software quality is characterized by certain attributes, which are highlighted by some standards. An example of such standard is ISO/IEC 9126. Main characteristics of ISO/IEC 9126 are functionality, maintainability, usability, efficiency, reliability, and portability [1], [2]. Software development industry has forced to place much more emphasis on software quality. In turn, researchers and practitioners have proposed a large number of new measures and assessment frameworks for quality design principles such as cohesion and coupling. High quality of any software means high cohesion. In object-oriented software, cohesion is the degree to which the methods and attributes of a class belong together.

Aspect-Oriented Programming (AOP) [3] languages aim to improve the ability of designers to modularize concerns that cannot be modularized using traditional module-oriented (MO) or object-oriented (OO) methods [4]. Such concerns are known as crosscutting concerns. Examples of crosscutting concerns include tracing, logging, caching and resource pooling. The ability to modularize such concerns is expected to improve comprehensibility, parallel development, reuse and ease of change [5], [6], reducing development costs, increasing dependability and adaptability. Since AO is a new abstraction, the definition of cohesion is required to redefine in the context of AOP.

The most popular AOP model today is of AspectJ [7]. AspectJ extends Java with several complementary mechanisms, namely join points (JPs), pointcut descriptors (PCDs), advice,
In section 5, applications of proposed framework are discussed. Conclusions and future work is in section 6.

2. Review of Cohesion Measures and Measurement Frameworks

There are many assessment framework and measurement approaches for cohesion measurement in OO Systems. It is because of high maturity level of OO paradigm. AOP is a very new and emerging paradigm. In literature, there exist few assessment framework/measurement approach/metrics [11], [12], [13], [14], [15] to measure cohesion in AO systems. However, most of the measures are for AspectJ programming language. There is no generic cohesion measurement framework.

2.1 Approach by Zhao and Xu [12]

First approach towards measuring aspect cohesion was by Zhao and Xu. Their approach is based on a dependency model for aspect-oriented software that consists of a group of dependency graphs. They defined cohesion as the degree of relatedness between attributes and modules (method/advice). They present, in fact, two ways for measuring aspect cohesion based on inter-attributes ($\gamma_a$), inter-modules ($\gamma_m$) and module-attribute ($\gamma_{ma}$) dependencies. The first way suggests that each measurement ($\gamma_a$, $\gamma_m$, $\gamma_{ma}$) works as a field. Therefore, aspect cohesion for a given aspect $A$ is defined as a 3-tuples $\Gamma(A) = (\gamma_a, \gamma_m, \gamma_{ma})$. They also suggested another way for cohesion measurement where each facet could be integrated as a whole with $\beta$ parameters. Aspect cohesion is then expressed as:

$$\Gamma(A) = \begin{cases} 0, & n = 0 \\ \beta^a\gamma_m, & k = 0 \text{ and } n \neq 0 \\ \beta_1^*\gamma_a + \beta_2^*\gamma_m + \beta_3^*\gamma_{ma}, & \text{others} \end{cases}$$

Where $k$ is the number of attributes and $n$ is the number of modules in aspect $A$. $\beta_i \in (0,1)$, $\beta_1, \beta_2, \beta_3 > 0$, and $\beta_1 + \beta_2 + \beta_3 = 1$.

Users determine the selection of weight $\beta_1, \beta_2$, and $\beta_3$, which is arbitrary. In addition, some relationships definition (inter-attributes cohesion in particular) and their considerations in aspect cohesion measurement are difficult to capture. In general, this approach suggests a complex way to measure aspect cohesion that may be problematic to use in a real development context and particularly in the case of real scalable aspect-oriented software. Generating such dependency graphs is a time consuming process while parameter’s weighting could be misleading. Secondary, using this approach, cohesion measurement is only for the family of AspectJ programming languages.
2.2 Framework by Sant’Anna, Garcia, Chavez, Lucena & Staa [13]
Sant’ Anna et al. proposed a new metric LCOO (Lack of Cohesion in Operations) measures the amount of method/advice pairs that do not access to the same instance variables. It is an extension of the well-known LCOM (Lack of Cohesion in Methods) metric developed by Chidamber and Kemerer [19]. This metric measures the lack of cohesion of a component (class and aspect). According to their approach, consider a component C1 with operations (methods and advice) Oi, ..., On. Let \( \{i_j\} \) = set of instance variables used by operation Oj. There are n such sets \( \{i_1\}, \{i_2\}, \ldots, \{i_n\} \). Let

\[
|P| = |\{i_1, i_2\} | i_1 \cap i_2 = \emptyset \}, \text{ and}
\]

\[
|Q| = |\{i_1, i_2\} | i_1 \cap i_2 \neq \emptyset \}
\]

If all n sets \( \{i_1\}, \ldots, \{i_n\} \) are \( \emptyset \) then let \( P = Q \).

LCOO = \[
\begin{cases} 
|P| - |Q|, & \text{if } |P| > |Q| \\
0, & \text{otherwise}
\end{cases}
\]

A high LCOO value, according to Sant’ Anna et al., indicates disparateness in the functionality provided by the aspect. The definition of LCOO metric is almost operational. It is not stated whether inherited operations and attributes are included or not, and we have to assume that sets \( \{i_j\} \) only include attributes of component C1. It suffers from several problems as stated, among others, by B. Henderson-Sellers in [20]. Again, definition of this metric is based on AspectJ-like programming languages.

2.3 Approach by Gelinas, Badri, and Badri [14]
Gelinas et al. proposed a new metric ACoh to measure aspect cohesion. This cohesion measurement is based on dependency analysis. They defined this metric using two aspect cohesion criteria, Modules-Data Connection Criterion and Modules-Modules Connection Criterion.

2.3.1 Modules-Data Connection Criterion
Let \( U_{A_{M_i}} \) be the set of attributes used directly or indirectly by the module \( M_i \). An attribute is used directly by a module \( M_i \) if the attribute appears in its body. An attribute is indirectly used by a module \( M_i \) if it is used directly by another module invoked directly or indirectly by \( M_i \). There are m sets \( U_{A_{M_1}}, U_{A_{M_2}}, \ldots, U_{A_{M_m}} \). Two modules \( M_i \) and \( M_j \) of an aspect are related by the \( U_{A_{M_i}} \) relationship if \( U_{A_{M_i}} \cap U_{A_{M_j}} \neq \emptyset \). It means that there is at least one attribute shared (directly or indirectly) by the two modules.

2.3.2 Modules-Modules Connection Criterion
Let \( U_{M_{M_i}} \) be the set of modules used directly or indirectly by the module \( M_i \). A module \( M_j \) is used directly by a module \( M_i \) if \( M_j \) appears in the body of \( M_i \). A module \( M_j \) is indirectly used by a module \( M_i \) if it is used directly by another module or indirectly by \( M_i \). There are m sets \( U_{M_{M_1}}, U_{M_{M_2}}, \ldots U_{M_{M_m}} \). Two modules \( M_i \) and \( M_j \) of an aspect are related by the \( U_{M_{M_i}} \) relation if \( U_{M_{M_i}} \cap U_{M_{M_j}} \neq \emptyset \). It means that there is at least one module jointly used (directly or indirectly) by the two modules. They also consider that \( M_i \) and \( M_j \) are directly related if \( M_j \in U_{M_{M_i}} \) or \( M_i \in U_{M_{M_j}} \).

2.3.3 Aspect Cohesion Measurement
Two modules \( M_i \) and \( M_j \) can be connected in many ways: by sharing attributes or by sharing modules or both. Let \( N(M(M_i)) \) be the total number of modules pairs in an aspect. \( N(M(M_i)) \) is the maximum number of connections between aspect modules. Thus in an aspect having N modules, \( N(M(M_i)) = \frac{N(N-1)}{2}, N>1 \). NC(Aspecti) be the number of connection between modules in undirected graph \( G_D \). They defined a new metric for aspect cohesion measures:

\[
ACoh(Aspecti) = \frac{NC(Aspecti)}{NM(Aspecti)} \in [0, 1].
\]

This cohesion measure metric is based on AspectJ-like programming languages. It is also not clear, whether they have accounted or not, inherited member and constructors.

2.4 A Framework by Sant’Anna, Figueiredo, Garcia and Lucena [15]
In this framework Sant’Anna et al. defined a new metrics suite for concern-driven architecture. The proposed metrics suite is to measure cohesion, coupling and complexity of AO and non-AO systems. The proposed metric LCC (Lack of Concern-based Cohesion) for measuring cohesion in concern-driven architecture counts the number of concerns addressed by the assessed component. This is very useful framework for evaluating internal software quality attributes such as cohesion, coupling, complexity and modularity at architecture level. It is not specified, how to quantify aspect cohesion for empirical evaluation. It is also not clear whether inheritance has been accounted or not.

3. A unified framework for cohesion measurement
To define a unified cohesion measurement framework, we have analyzed frameworks for OO systems [17], AO systems [12], [13], and specific for AspectJ [14]. Also we have identified the influences of CaesarJ and it’s specific features. From these analyses, a list of criterion that must be considered when proposing a new cohesion measurement framework emerged. These criteria are supposed to answer the questions. The cohesion criteria that we have considered in this proposed framework are originally defined in Briand’s [14] framework.

Our framework consists of five criterions; each criterion determines one basic feature of the resulting measure. First we describe each criterion, what decisions have to be made, what are the available options, how the criterion reflected by cohesion measures, and how the framework can be used to derive cohesion measures.
3.1 Type of connections
What constitutes cohesion?
This criterion defines mechanism that causes cohesion in component. A connection within a component is a link between elements of the component (operation, attribute, inter-type attribute, or data declarations). In this framework we take into account connections related to OO cohesion [17], AspectJ specific connections [14], and generic AO connections [12]. We also study possible connections for cohesion caused by CaesarJ type constructs.

Definition 13: Type of connection of a component $c$
Let operations $o, o' \in O(c)$ and an attribute $a \in A(c)$

- CON#1: $o$ references $a$.
- CON#2: $o$ invokes $o'$.
- CON#3: $o$ and $o'$ directly reference $a$ of $c$ in common ("similar operations").
- CON#4: $o$ and $o'$ directly or indirectly reference $a$ of $c$ in common ("connected operations").
- CON#5: data-data interaction (data declaration in $c$)
- CON#6: data-operation interaction

Where $CON#$, represents connection number.

3.2 Domain of measure
Which element to account for, which to restrict?

By domain of measure means finding out which elements (attribute, operation, etc.) will participate in measuring cohesion of the component. Most of the reviewed measures are defined at the component level. However, finer and coarser domains are also conceivable. We have also considered new elements (eclass, wrapper constructor etc.) of CaesarJ family. We can count the number of other component elements to which it is connected for an individual attribute or operation. This is analyzing how closely related the attribute or operation is to other elements of its component. In other words it is the degree to which the attribute or operation contributes to the cohesion of its component. From this measure, we can analyze which element of the component should remain in the same component and which should be moved to new or another component.

3.3 Direct or indirect connections
Count direct connections only or also indirect connections?
In this section our focus is whether to count direct connections only or also indirect connections. Direct connections definitely make a component cohesive (All connection types CON#1 to CON#6). But for indirect connection cohesion measures depend on the type of connections. For example, consider an operation $o_1$, which is “similar” to another operation $o_2$ (CON#3), operation $o_2$ is “similar” to operation $o_3$ (CON#3).

Operations $o_1$ and $o_2$ are directly connected. Similarly operations $o_2$ and $o_3$ are also directly connected. Now we can say operations $o_1$ and $o_3$ are indirectly connected, which is a transitive relationship connection.

3.4 Inheritance
Whether to account or not the members of Parents($c$), and Ancestors($c$) of component $c$?

There are two different aspects to be considered with respect to inheritance.

- How do we assign attributes and operations to components?
- Whether to consider static or polymorphic invocation of operation?

These could be described as:

How do we assign attributes and operations to components?

For the analysis of cohesion measures, there are two elements of a component $c$, attributes $A(c)$ and operations $O(c)$. For the measures of cohesion, either we can account non inherited elements only or also account inherited elements.

1. Exclude inherited attributes and operations from the analysis.

If we exclude inherited attributes and operations, cohesion analysis will be to the same component only and will represent a single semantic concept.

2. Include inherited attributes and operations in the analysis.

If we include inherited attributes and operations, cohesion measures accounts attributes inherited, attributes newly added and operations in the component $c$. We will analyze whether component $c$ as a whole still represents a single semantic concept.

Whether to consider static or polymorphic invocation of operation?

Polymorphism will be relevant only if the type of connection involves operation invocation. If $c$ is a component, and Ancestors($c$) is/are the ancestor(s) of $c$, then we have following two options:

1. Do not account for polymorphism. Let $o$ be an operation of component $c$, which is having connection with operations $o'$. $o'$ can possibly be invoked statically in the implementation of $o$.

2. Account for polymorphism. Let $o$ be an operation of the component $c$, which is having connections with operations $o'$. $o'$ can possibly be in the implementation of $o$ through polymorphism and dynamic binding.

3.5 Access operations and constructors
How to account for access operations and constructors for measuring cohesion?

Access operations and constructors may artificially increase or decrease cohesion measure values. In the definition of cohesion
measures and measurement framework, we have to decide how to account access operations and constructors.

**Access operations**

Following options can be used to account for access operations

1. Treat access operations as regular operations  
   Do nothing; consider access operations as regular operations, which may cause artificially increase or decrease in cohesion values.

2. Consider the invocation of an access operation as a reference to the attribute  
   If an operation is not referring an attribute directly, an access operation could be used to refer the attribute indirectly. These access methods are not accounted for by CON#1 & CON#3, which causes artificially decreased number of references to attributes. If we account the invocation of access operation as reference to the attributes, it will be solution to this problem. Implementation of this problem is difficult, because to recognize access operations automatically is not always possible.

3. Exclude the access operations from the analysis  
   If pair of operations are using common attributes (CON#3 & CON#4), then access operations may cause problems for cohesion measures. Access operations usually access only one attribute, many pairs of operations that don’t refer a common attribute can be found using access operation. This problem causes artificially decrease in cohesion. Solution to this problem is to exclude access operations from the analysis.

**Constructors**

Although we have considered constructors (All) and Wrapper Constructors (CJ) as part of operation in section 2.4, even then it is important to discuss effects of constructors and wrapper constructors separately. Either we can include constructors in cohesion analysis or we can exclude constructors from it.

1. Include constructors in the analysis  
   Do nothing, count constructors and wrapper constructors as operation like method and advice.

2. Exclude constructors from the analysis  
   Constructors and wrapper constructors cause problems for measures that count pairs of operations which use common attributes (CON#3 and CON#4). Most of the time constructors typically reference all the attributes. It causes artificially increased in cohesion of the component, because it generates many pairs of operations that use a common attribute. The solution to this problem is to exclude constructors from the analysis.

4. CONCLUSION

Cohesion is one of the important internal software attributes that has been both theoretically and empirically shown to have large impact on external software quality characteristics like maintainability, modularity, reusability, testability and usability. Aspect-oriented programming is a new and emerging paradigm. Quality assessment works are in initial stages in aspect-oriented systems. Cohesion measures, definition and implementation of metrics for cohesion and cohesion measurement frameworks for AO systems are also in premature stage. Even though, many researchers and practitioners have proposed cohesion metrics, cohesion measures and cohesion measurement frameworks for AOP. We have reviewed the existing measures and framework on the basis of cohesion measurement criterions. Most of the existing measures are for AspectJ-like languages. There is a pressing need to propose and develop cohesion measurement frameworks for other families of aspect-oriented programming languages such as CaesarJ, Hyper/J. There is also a pressing need to propose and validate a generic/unified cohesion measurement framework which should account most of the popular AOP languages.

**FUTURE SCOPE**

In the continuities of this work, we are developing a unified cohesion measurement framework, which is accounting Java, AspectJ-like languages, family of CaesarJ AOP languages and Hyper/J. This proposed cohesion measurement framework will help in (i) definition of new cohesion metrics (ii) integration of different measures and measurement frameworks (iii) proper understanding of cohesion in different prospective.

5. REFERENCES


