

An Evaluation of Coupling Measures for AspectJ

Marc Bartsch
School of Systems Engineering
The University of Reading, Reading
RG6 6AY, UK
+44(0)118 378 8617
m.bartsch@reading.ac.uk

Rachel Harrison
School of Systems Engineering
The University of Reading, Reading
RG6 6AY, UK
+44(0)118 378 8617
rachel.harrison@reading.ac.uk

ABSTRACT

The maintenance of aspect-oriented software requires measures that are theoretically valid. Inadequately validated measures might lead to management or project decisions that are likely to be not as effective as they could be. Recently, measures have been suggested that focus on aspect-oriented concepts, such as the crosscutting behaviour of aspects. Before these new measures can be put to use they should be evaluated to determine how far they indeed measure what they purport to quantify. This paper focuses on the evaluation of five aspect-oriented coupling measures with the aim to constructively increase the quality of software evolution.

Categories and Subject Descriptors

D.2.8 [Software Engineering]: Measurement – *product metrics*.

General Terms

Measurement.

Keywords

Measurement, aspect-oriented, coupling.

1. INTRODUCTION

Aspect-orientation is an emerging paradigm that is based on the separation of concerns principle. It offers the idea of a new modular unit that encapsulates crosscutting concerns which would otherwise be scattered across multiple modules. Aspect-oriented languages introduce new forms of coupling which are unknown to object-oriented languages. The execution of base code might trigger the execution of aspect code leading to coupling relationships between classes and aspects which are not transparent. In addition to that, intertype declarations can change class implementations by adding attributes or methods.

Measuring these new kinds of coupling relationships is an issue which has been addressed lately with the definition of coupling measures specifically designed to support aspect-oriented

concepts. A maintenance process for aspect-oriented software that relies on the results of these measures must have the confidence that the measures involved indeed measure what they purport to quantify. Also, the comparison of measurement results is an error prone task if measures can be interpreted in different ways.

The position of the authors is that all measures including coupling measures need to be validated to gain confidence in the results taken from measurement. However, others have pointed out that metrics that cannot be validated may still be useful [6]. Since research into measurement of aspect-oriented systems is at an early stage, it is particularly important to validate aspect-oriented measures thoroughly. Also, the authors would like to stress the fact, that validating aspect-oriented coupling measures depends on at least two frameworks. First, validation criteria need to be agreed upon. A measure that validates successfully in the context of one framework might not validate in another. Second, aspect-oriented coupling measures depend on a specific idea of coupling. Mechanisms that constitute coupling in one aspect-oriented language might not exist in another. Hence, a specific aspect-oriented language implementation has to be considered, when aspect-oriented coupling measures are validated.

A first step in this direction is the evaluation of five selected aspect-oriented coupling measures suggested by Ceccato and Tonella [4]. In the evaluation process we will focus on four points: First we give an overview of the suggested coupling measures. Second, we identify a measurement goal and a rationale behind the measures. When measures are to be selected for a certain measurement goal, this selection process should be supported by the measure. Third, we ask whether the definitions are well-defined, i.e. whether they are free of any ambiguities. Last, we will investigate whether the coupling measures can be considered valid from a measurement theory point of view, i.e. whether their definitions conform to criteria that all valid measures must obey. Without such a validation we can have no confidence in whether they indeed measure what they purport to.

This paper is structured as follows: section 2 presents related work, section 3 introduces two evaluation frameworks used in this paper, section 4 introduces and evaluates the coupling measures. Finally, section 5 offers conclusion and points to further research.

2. RELATED WORK

Aspect-oriented measures derived from Chidamber and Kemerer's suite of object-oriented measures [5] have been suggested by Ceccato and Tonella [4] and by Sant'Anna et al. [9]. Zhao defines coupling measures for aspect-oriented systems [10] and cohesion measures [11].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

LATE'06, March 20, 2006, Bonn, Germany.

Copyright 2006 ACM 0-00000-000-0/00/0000...\$5.00.

The issue of validating measures has been approached from different viewpoints. In [3], Briand et al. suggest mathematical criteria that all coupling measures must conform to. Kitchenham et al. define a set of criteria that all measures must obey to be considered a valid measure from a measurement theory point of view [8]. This framework has been used in this paper to validate the coupling measures. In [7], Kaner and Bond define a list of 10 questions that they consider relevant from a practical viewpoint.

3. EVALUATION FRAMEWORKS

The evaluation process of aspect-oriented coupling measures will be guided by two frameworks. First, a coupling framework for AspectJ will be used that offers a common terminology to express the different coupling measures in a uniform way [2]. Also, the criteria of the framework give a guideline for evaluation and comparison. In particular, the coupling framework is a means to find out about ambiguities associated with the algorithm of the different measures.

Second, a validation framework by Kitchenham et al. is used for the purpose of a theoretical validation of each measure [8]. The validation framework consists of four criteria that all valid measures must obey. We will use this framework to validate each measure against the validation criteria.

3.1 Coupling Framework

Bartsch and Harrison [2] suggest a coupling framework for AspectJ which is an extension of a coupling framework for object-oriented systems put forward by Briand et al. [3]. The extended framework contains a specific definition of the different coupling mechanisms found in AspectJ [1] and an additional criterion: instantiation. The criteria of the framework are:

1. *Type of Connection.* The type of a particular coupling connection is determined by the mechanism that is used to establish the coupling connection. The use of a class identifier as a return type of a piece of advice, for example, is a mechanism that leads to coupling between the aspect that implements the advice and the class whose identifier has been used. The adapted coupling framework focuses on coupling mechanisms that do not exist in object-oriented languages or that exists between members which are unknown in object-oriented languages.
2. *Locus of Impact.* If an aspect is used in a coupling connection, a distinction is made between import and export coupling which defines a counting rule for a coupling connection. If aspects invoke methods of other classes, then import coupling counts the number of classes whose methods would be called by a given aspect. Export coupling counts the number of aspects that make calls to a method of a given class.
3. *Granularity.* Granularity refers to the level of detail at which coupling information is collected. It indicates the components that are counted and how to count multiple occurrences of a connection.
4. *Stability of Server.* Stability of server expresses whether components at the receiving end of a coupling connection are subject to modifications and might influence coupled classes if modifications are applied. This criterion is independent from the distinction between object-oriented or aspect-

oriented languages and is beyond the scope of this paper. For the remainder of this paper, only stable classes and aspects will be considered.

5. *Direct or Indirect Connections.* Direct or indirect connections refer to whether to count direct coupling connections only or also indirect connections. For the remainder of this paper, only direct connections will be considered.
6. *Inheritance.* The following decisions have to be made with regard to inheritance: First, does the use of members of an ancestor class constitute coupling or not? Second, does a measure consider both polymorphically and statically invoked methods and, third, do inherited members belong to the inheriting aspect or not?
7. *Instantiation.* Instantiation refers to the question whether to count aspects at a per-instance level or not.

Originally, the definitions of the measures investigated in this paper focused on the first three criteria only, i.e. they primarily addressed types of connections, the locus of impact and granularity. For those criteria not addressed in the measures, we assume default values as defined in the framework above. Ambiguities that could arise from a lack of clarity as far as these criteria are concerned will be discussed in section 5. In the evaluation process we will therefore focus on the first three criteria only.

3.2 Validation Framework

Kitchenham et al. propose a set of four criteria that all valid measures must obey [8]:

1. For an attribute to be measurable, it must allow different entities to be distinguished from each other.
2. A valid measure must obey the Representation Condition [6], i.e. it must preserve our intuitive notion about the attribute and the way in which it distinguishes different entities.
3. Each unit of an attribute contributing to a valid measure is equivalent.
4. Different entities can have the same attribute value (within the limits of measurement).

In this paper, we will focus on criterion 2, the Representation Condition. Since all coupling measures which we will discuss are based on counting coupling connections only, criteria 1, 3 and 4 can be validated intuitively and will not be considered any further.

Notions about attributes are subjective and can vary. Not everyone might agree upon what constitutes coupling in aspect-oriented systems or in a particular language like AspectJ. In order to validate a measure against the Representation Condition, first, the notion of the measure has to be defined. Second, an algorithm needs to be specified that defines how the values of the measure will be computed. The validation process consists of verifying that the values support the agreed notions about the measure. The Representation Condition is also highlighted during consideration of Construct Validity [7]. Construct Validity generalises from a measure or the algorithm of a measure to the concept of it, and asks whether the algorithm really captures the notion of a certain

measure. The process of validation is then an investigation into how far the algorithm agrees with the notion being intended.

An operational definition is the description of the process that leads to the quantification of a certain attribute. An operational definition is unambiguous and self-contained, i.e. an operational definition contains all the information that is needed to measure an attribute. When validating measures, an operational definition is helpful, since only then is the stated algorithm well-defined. For our purpose, however, we will use a weaker form of operational definition and require our measure to be unambiguous according to the coupling framework introduced above. For example, we require the measures under investigation to be explicit about the coupling mechanisms they employ or the counting rules that they use. Only through this mapping of the measures' original definition onto our coupling framework can comparability and validation be achieved.

4. EVALUATION

As a result of the discussion in section 3, the evaluation process will include the following steps for each measure. First, the notion of the measure will be presented. Second, the original algorithm will be stated and third, each algorithm will be translated into our coupling framework. The process of translation will reveal ambiguities in the definition for which adjustments will be suggested, if necessary. Last, each measure will be validated by an investigation into the Representation Condition, i.e. an investigation into the notion and the algorithm involved. Mostly, this will involve the question whether a certain measure considers all necessary coupling mechanisms that can be associated with a certain notion of coupling.

Well-definedness and validation of a measure can be related. If a measure is not well-defined, i.e. if its definition contains ambiguities, then this measure cannot be validated successfully since the algorithm for deriving values might be interpreted in different ways and might lead to different values. If a measure is well-defined, it may still not be valid in terms of the validation framework. Even if an algorithm contains no ambiguities, it might not cover all coupling mechanisms that are related to this particular notion of coupling. Such a measure violates the Representation Condition since results from this measure cannot distinguish between programs that are identical except for the use of coupling mechanisms not considered by the measures. It would yield the same values even though, intuitively, such programs may contain different degrees of coupling.

Ceccato and Tonella define five coupling measures in order to investigate the trade-off between the advantages obtained from a separation of concerns and the disadvantages caused by coupling introduced by aspects. The target design level is low-level design since the measures take into account coupling mechanisms which are only fully available at this design stage.

4.1 CAE (Coupling on Advice Execution)

Notion. Aspects can cause control flow shifts so that advice is executed in the course of a program's execution. CAE [4] quantifies the amount of coupling caused by these shifts for a given class or aspect.

Algorithm. CAE counts the number of aspects containing advices possibly triggered by the execution of methods, advices or method intertype declarations in a given class or aspect.

Well-Definedness. The measure uses selected join point coupling mechanisms, i.e. all mechanisms that refer to executions: *method*, *constructor* or *advice execution join point coupling*. The granularity is *aspect* and every aspect will be counted only once. The locus of impact is *export coupling*. However, the algorithm includes the phrase "possibly triggered" which is not sufficiently defined. It might hint at the *cf flow pointcut designator* which only allows an exact determination of aspect execution at runtime. Therefore, this measure exhibits at least one ambiguity and cannot be adequately mapped to the coupling framework.

Validation. This measure cannot be validated for two reasons. First, the ambiguity of the measures algorithm is difficult to resolve. Second, the notion of CAE focuses on the execution of advice. Such an execution can be caused not only by the execution of methods, advice or method intertype declarations, but also by the access of attributes. This measure needs to address all types of join point coupling mechanisms to be a measure of coupling caused by advice execution.

4.2 CIM (Coupling on Intercepted Modules)

Notion. CIM [4] quantifies the explicit knowledge that an aspect has in its pointcuts of another class or interface that it crosscuts. It indicates the direct knowledge an aspect has of the rest of the system. High values indicate high coupling, due to high crosscutting.

Algorithm. CIM counts the number of classes or interfaces explicitly named in the pointcuts of a given aspect.

Well-Definedness. This definition can be mapped unambiguously to the criteria of the coupling framework. The type of connection is *type pattern coupling*, the granularity is *class* or *interface* and every class or interface will be counted once. The locus of impact is *import coupling*.

Validation. CIM cannot be validated successfully. This measure makes the assumption that an explicitly mentioned class or interface in a pointcut always leads to advice being executed, i.e. to crosscutting. This is not always the case. Consider the following example:

```
pointcut userMethods() : execution( void *.add() )
                        && ! execution( void Manager.add() );
```

Figure 1. Type pattern based coupling.

Although the *Manager* class is explicitly mentioned in the pointcut, it is excluded from the set of join points. Hence, the aspect containing this pointcut does not crosscut the *Manager* class and leads to the execution of advice. A minor problem is that the mere definition of a pointcut is not enough to crosscut base code. It also needs to be referred to in at least one piece of advice.

The problems with this measure can be solved in two different ways. First, the notion (and the name) can be altered so that CIM only quantifies the explicit knowledge an aspect has of the rest of the system, regardless whether this knowledge leads to

crosscutting. Second, the algorithm could be altered to count only explicitly mentioned classes or interfaces that in fact contribute to crosscutting and that are referred to in at least one piece of advice.

4.3 CMC (Coupling on Method Call)

Notion. CMC [4] is based on Chidamber and Kemerer’s CBO measure. CMC focuses on method calls only and quantifies the amount of coupling due to the call of methods or method intertype declarations of other classes or aspects.

Algorithm. CMC counts the number of classes or aspects that declare methods which are possibly called by a given class or aspect, including those methods that have been introduced by intertype declarations.

Well-Definedness. The type of connection is *method execution*, the granularity is *class* or *aspect* and every class or aspect will be counted once. However, this definition cannot be mapped unambiguously to the criteria of the coupling framework. The locus of impact is not clear. CBO counts *import* and *export coupling* while the algorithm for this measure only states *import coupling*. Also, the algorithm includes the phrase “possibly called” which is not further defined.

Validation. CMC cannot be validated due to the ambiguities in its algorithm. It does, however, collect all the coupling mechanisms that contribute to its notion.

4.4 CFA (Coupling on Field Access)

Notion. CFA [4] is designed similarly to CMC. Where CMC measures coupling due to method calls, CFA measures coupling due to attribute accesses.

Algorithm. CFA counts the number of classes, interfaces or aspects declaring attributes that are accessed by a given class or aspect.

Well-Definedness. The type of connection is *attribute access*, the granularity is *class*, *interface* or *aspect* and every class, interface or aspect will be counted only once. Just like CMC, this measure is unclear about the locus of impact, *import* or *export coupling*, and, thus, cannot be mapped unambiguously to the framework.

Validation. As for CMC, this measure cannot be validated without a clear definition of the locus of impact. Other than that, the algorithm captures the notion of coupling set out above.

4.5 CDA (Crosscutting Degree of an Aspect)

Notion. CDA [4] captures the overall crosscutting impact that an aspect has on the system through pointcuts and intertype declarations. The difference between CDA and CIM gives the number of modules that are affected by an aspect without being referenced explicitly by the aspect.

Algorithm. CDA counts the number of classes or aspects affected by the pointcuts and by the intertype declarations of a given aspect.

Well-Definedness. The locus of impact is *import coupling*, the granularity is *class* or *aspect* and every class or aspect will be counted only once. This definition cannot be mapped unambiguously to the criteria of the coupling framework since the term “possibly affected” is not sufficiently defined. We interpret

the type of connections to be all classes that can be associated with join points which lead to the execution of advice. Also, CDA seems to consider explicitly mentioned classes and interfaces and all classes that are extended by intertype declarations. However, this is merely our interpretation and needs further clarification. As far as the difference between CDA and CIM is concerned, it does not indicate the number of modules that are affected by an aspect without being referenced explicitly. Since CDA considers intertype declarations, which CIM neglects, it might count modules which do not appear in pointcut expressions but which are nonetheless referenced explicitly. The degree of generality intended by CDA can only be maintained if it considers modules affected by pointcuts only and not by intertype declarations.

Validation. CDA cannot be validated successfully without clarification of its algorithm.

4.6 Summary

The following table gives an overview of the measures and their mapping and validation status:

Table 1. Overview of all measures

Measure	Well-Defined	Validated
CAE	No	No
CIM	Yes	No
CMC	No	No
CFA	No	No
CDA	No	No

All of the investigated measures have clearly defined notions, i.e. they refer to the attribute which they quantify and give a rationale that explains their design.

Only one measure, CIM, can be mapped without ambiguities to the coupling framework. CAE, CMC and CDA require clarification with regard to the use of “possibly” or “possibly affected” to determine exactly which coupling connections to count. Without such a clarification, the measures could be interpreted in different ways. A similar problem can be found with CMC and CFA as they do not define the locus of impact explicitly.

The coupling framework consists of a number of other criteria that have not been addressed by any of the measures and have not been discussed so far. These criteria might be the source of further ambiguities, in particular inheritance. Two of the measures are based on counting method invocations or attribute accesses (CMC, CFA) but it has not been stated clearly how far inheritance influences the determination of those attributes and methods. Default values could be assumed but if the same measure is used in a different context, this assumption might not apply. Thus, all measures that potentially suffer from a lack of clarity in one of the areas should explicitly define the influence of inheritance. Such an investigation is beyond the scope of this paper but will be addressed in future work.

Although none of the measures could be validated in the context of the validation framework used, most do not exhibit any major problems. CAE, CMC and CFA can be adapted easily to conform to the validation framework if the ambiguities of their algorithms

have been resolved. CIM and CDA also can be validated if their algorithms have been adjusted to the notions intended.

5. CONCLUSION

Overall, it has been shown that the quality of most of the measures considered here can be increased by more clarity in their definitions. Our conclusion is that the definition of coupling measures is a difficult task that needs a great deal of rigour if the measures are to be theoretically valid and well-defined. This can only be a first step to ensure that empirical validation operates on measures that are in fact designed for their measurement purpose.

Further research will include further research into the validity of other sets of aspect-oriented coupling measures and tool support to collect values for single coupling connections. Such a tool can be used to investigate dependencies among different coupling mechanisms and will help to point towards the definition of coupling measures that are likely to predict qualities such as maintainability.

6. REFERENCES

- [1] AspectJ, 2005: <http://www.eclipse.org/aspectj>
- [2] Bartsch, M. and Harrison, R., 'A Coupling Framework for AspectJ', <http://www.personal.rdg.ac.uk/~sir04mb2/Publications/bartsch-harrison-couplingFramework.pdf>.
- [3] Briand, L.C., Daly, J.W. and Wüst, J.K. (1999) A Unified Framework for Coupling Measurement in Object-Oriented Systems. *IEEE Transactions on Software Engineering*, 25(1), 91-120.
- [4] Ceccato, M. and Tonella, P., 'Measuring the Effects of Software Aspectization', *CD-Rom Proceedings of the 1st Workshop on Aspect Reverse Engineering (WARE 2004)*, Nov. 2004, Delft, The Netherlands.
- [5] Chidamber, R. and Kemerer, C.F., 'A Metrics Suite for Object-Oriented Design', *IEEE Transactions on Software Engineering*, 20(6):476-493, 1994.
- [6] Fenton, N.E. and Pfleeger, S.L., '*Software Metrics: A Rigorous and Practical Approach (2nd Edition)*', International Thomson Computer Press, 1996.
- [7] Kaner, C. and Bond, P., 'Software Engineering Metrics: What Do They Measure and How Do We Know?', *10th International Software Metrics Symposium (Metrics 2004)*, Chicago, IL, September 14-16, 2004.
- [8] Kitchenham, B.A., Fenton, N. and Pfleeger, S.L., 'Towards a Framework for Software Measurement Validation', *IEEE Transactions on Software Engineering*, 1995, Vol.21, No.12, pp. 929-944.
- [9] Sant'Anna, C., Garcia, A., Chavez, Ch., Lucena, C. and von Staa, A., 'On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework', *Proceedings of the Brazilian Symposium on Software Engineering (SBES'03)*, Manaus, Brazil, Oct 2003, pp.19-34.
- [10] Zhao, J., 'Measuring Coupling in Aspect-Oriented Systems', *10th International Software Metrics Symposium (METRICS'2004)*, (Late Breaking Paper), Chicago, USA, Sept. 14-16, 2004.
- [11] Zhao, J. and Xu, B., 'Measuring aspect cohesion'. *Proceedings of the International Conference on Fundamental Approaches to Software Engineering (FASE)*, LNCS 2984, Barcelona, Spain, March 2004. Springer Verlag, pp.54-68.