# Coupling Metrics for Aspect-Oriented Programming: A Systematic Review of Maintainability Studies (Extended Version)

Rachel Burrows[1], Alessandro Garcia[2], François Taïani[1]

[1] Computing Department, Lancaster University, UK
{rachel.burrows, francois.taiani}@comp.lancs.ac.uk
[2] Informatics Department, Pontifical Catholic University of Rio de Janeiro, Brazil
afgarcia@inf.puc-rio.br

**Abstract.** Over the last few years, a growing number of studies have explored how *Aspect-Oriented Programming* (AOP) might impact software maintainability. Most of the studies use coupling metrics to assess the impact of AOP mechanisms on maintainability attributes such as design stability. Unfortunately, the use of such metrics is fraught with dangers, which have so far not been thoroughly investigated. To clarify this problem, this paper presents a systematic review of recent AOP maintainability studies. We look at attributes most frequently used as indicators of maintainability in current aspect-oriented (AO) programs; we investigate whether coupling metrics are an effective surrogate to measure theses attributes; we study the extent to which AOP abstractions and mechanisms are covered by used coupling metrics; and we analyse whether AO coupling metrics meet popular theoretical validation criteria. Our review consolidates data from recent research results, highlights circumstances when the applied coupling measures are suitable to AO programs and draws attention to deficiencies where coupling metrics need to be improved.

**Keywords:** Coupling, Aspect-Oriented Programming, Systematic Review, Maintainability

## 1 Introduction

*Aspect-oriented programming* (AOP)[2] is now well established in both academic and industrial circles, and is increasingly being adopted by designers of mainstream implementation frameworks (e.g. *JBoss* and *Spring*). AOP aims at improving the modularity and maintainability of crosscutting concerns (e.g. security, exception handling, caching) in complex software systems. It does so by allowing programmers to factor out these concerns into well-modularised entities (e.g. aspects and advices) that are then *woven* into the rest of the system using a range of composition mechanisms, from *pointcuts* and *advices*, to *inter-type declarations*[27], and *aspect collaboration interfaces*[8].

Unfortunately, and in spite of AOP's claims to modularity, it is widely acknowledged that AOP mechanisms introduce new intricate forms of coupling[33], which in turn might jeopardise maintainability[1,4]. To explore this, a growing number of exploratory studies have recently investigated how maintainability might be impacted by the new forms of coupling introduced by AOP mechanisms[e.g 19,20,26].

The metrics used by these studies are typically taken from the literature[10,11,33,37,39] and are assumed to effectively capture coupling phenomenon in AOP software. However, the use of coupling metrics is fraught with dangers, which as far as AOP maintainability is concerned have not yet been thoroughly investigated. In order to measure coupling effectively a metrics suite should fulfill a number of key requirements. For instance: the suite should take into account all the composition mechanisms offered by the targeted paradigm[29,31]; the metrics definitions should be formalised according to well-accepted validation frameworks, e.g. Kitchenham's validation framework[30]; and they should take into account important coupling dimensions, such as coupling type or strength. If these criteria are not fully satisfied, maintainability studies of AOP might draw artificial or inaccurate conclusion and, worse, might mislead programmers about the potential benefits and dangers of AOP mechanisms regarding software maintenance.

Unfortunately, the validity and reliability of coupling metrics as indicators of maintainability in AOP systems remains predominantly untested. In particular, there has been no *systematic review* on the use of coupling metrics in AOP maintainability studies. Inspired from medical research, a systematic review is a fundamental empirical instrument based on a literature analysis that seeks to identify flaws and research gaps in existing work by focusing on explicit research questions[29]. This paper proposes such a systematic review with the aim to pinpoint situations where existing coupling metrics have been (or not) effective as surrogate measures for key maintainability attributes. Our systematic review consolidates data from a range of relavent AOP studies, highlights circumstances when the applied coupling measures are suitable to AO programs and draws attention to deficiencies where coupling metrics needs to be improved.

The remainder of this paper provides some background on AOP and coupling measurement (Section 2). We then discuss the design of our systematic review and present its results (Section 3 and 4). Finally, we discuss our findings (Sections 5) and conclude (Section 6).

## 2 AOP and Coupling Measurement

This section gives a brief discussion on three representative AOP languages and also gives a background on coupling metrics for AOP.

### 2.1 AOP Languages and Constructs

One of the reasons why the impact of AOP on maintainability is difficult to study pertains to the inherent heterogeneity of aspect-oriented mechanisms and languages. Different AOP languages tend to incarnate distinct blends of AOP and use different encapsulation and composition mechanisms. They might also borrow abstractions and composition mechanisms from other programming paradigms, such as collaboration languages (CaesarJ), feature-oriented programming (CaesarJ), and subject-oriented programming (HyperJ). Most AOP languages tend to encompass conventional AOP properties such as joinpoint models, advice and aspects, or their equivalent, but each possesses unique features that make cross-language assessment difficult.

Table 1 lists ten such features for AspectJ[2], HyperJ[23] and CaesarJ[8], three of the most popular AOP languages. For instance, AspectJ supports advanced dynamic pointcut designators, such as "cflow". HyperJ uses hyperspace modules to modularise crosscutting behaviour as well as non-crosscutting behaviour. HyperJ thus does not distinguish explicitly between aspects and classes in the way AspectJ does. Other abstractions unique to HyperJ include Compositions Relationships. These use merge-like operators to define how surrounding modules should be assembled. Finally, CaesarJ supports the use of virtual classes to implement a more pluggable crosscutting behaviour. This pluggable behaviour is connected with the base code through Aspect Collaboration Interfaces.

**Table 1.** AO abstractions and mechanisms unique to three main AOP languages

| AO Language | Abstraction / Mechanism |
|---|---|
| AspectJ | Intertype Declaration |
| | Dynamic Pointcut Designators |
| | Aspect |
| CaesarJ | Aspect Collaboration Interface |
| | Weavlet |
| | Virtual Class |
| HyperJ | Hyperspace |
| | Concern Mapping |
| | Hypermodule |
| | Composition Relationship |

### 2.2 Existing AO Coupling Metrics

Coupling metrics aim to measure the level of interdependency between modules within a program[12], thus assessing a code's modularisation, and indirectly maintainability. Unfortunately, each language's unique features introduce new forms of coupling, which cannot always readily be mapped onto existing concepts (Table 1). This creates a challenge when designing coupling metrics for AOP, as these metrics should ideally take into account each language's unique features, while still providing a fair basis for comparison multiple AOP languages. This is particularly difficult.

A number of coupling metrics have so far been proposed for AO programs. Some are adapted from object-orientation, and transposed to account for AO mechanisms. For instance, both Ceccato and Tonella[10] and Sant'Anna et al[36] have proposed coupling metrics adapted from an object-oriented (OO) metrics suite by Chidamber and Kemerer [11]. These metrics can be applied to both OO and AO programs. This is especially useful in empirical studies that perform aspect-aware refactoring. Unfortunately, because these metrics are not specific to AOP, they might overlook the unique intricate forms of coupling described in Table 1.

Zhao[39] uses dependency graphs to measure some AO mechanisms that are not measured individually in either Ceccato and Tonella or Sant'Anna's suites. Zhao's suite contains metrics that measure coupling sourced from AO abstractions and mechanisms independently of OO abstractions and mechanisms.

Coupling metrics are however rarely used as a direct representation of maintainability, but instead are typically contrasted against a particular maintainability attribute, such as code stability. The choice of this attribute (or attributes) might in turn influence which coupling metrics is the most suitable.

## 3 Systematic Review

This section describes the objectives and questions (Section 3.1) as well as the strategical steps carried out in the systematic review.

### 3.1 Objectives and Questions

The *aim* of our systematic review is to analyse the effectiveness of coupling metrics in existing AO empirical studies as a predictor of maintainability, and in particular focus on the following four *research questions:*

a) Which external attributes are most frequently used to indicate maintainability in current AO programs?
b) Are used coupling metrics effective surrogate measures for software maintainability?
c) Are all AOP abstractions and mechanisms covered in the design of the used coupling metrics?
d) Do AO coupling metrics meet well-established theoretical validation criteria?

### 3.2 Review Strategy

Searches for papers took place in 14 renowned online journal banks or were those published in recognised conference papers such as AOSD(Aspect-Oriented Software Development) and ECOOP (European Conference on Object-Oriented Programming). We gave priorities to publications in conferences with an acceptance rate below 30%. Relavent papers were found from ACM, SpringerLink, IEEE, Google Scholar, Lancaster University Online Library, and two were collected from other sources.

*Sampling Criteria:* From this base we sampled papers that met the following criteria. Each selected paper had to:

- use an empirical study to measure maintainability attributes in AOP;
- and use coupling metrics within the study.

Due to low retrieval rate from journal banks, alternative approaches were also used. This included both consulting references on already-found papers and searching specifically for papers we knew met our criteria (from previous knowledge). The distribution of collected research is recorded in the review results (Section 4).

*Exracted Data*: We recorded which independent / dependent variables where measured, the goals of measurement, the type of study, measurement results, which coupling metrics were used, their origin, and whether the applied metrics were specifically for AOP or adapted from another programming technique (e.g. OOP).

## 4 Results

A final set of 12 papers was finally obtained (Table 2), which is a typical sample size for systematic reviews in software engineering[28].

**Table 2.** Distribution of Studies

| Electronic Journal | # Retrieved | # Rejected | # Used |
|---|---|---|---|
| ACM | 4 | 0 | 4 |
| IEEE | 2 | 1 | 1 |
| SpringerLink | 3 | 1 | 2 |
| L.U. Online Library | 5 | 2 | 3 |
| Other | 4 | 2 | 2 |
| **Total** | **18** | **6** | **12** |

### 4.1 Assessed Maintainability Attributes

It is difficult to select coupling metrics to assess maintainability as definitions are often open to interpretations. For instance in[24], maintainability is “the ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment”.

There is also no consensus about the external and internal attributes are the most significant indicators of maintainability. This is apparent in the empirical studies from the diverse selection of metrics used. Two main processes were recorded to select

suitable coupling metrics. Firstly, many studies used coupling metrics previously selected in similar AOP empirical studies. Secondly, results showed the Goal-Question-Metric (GQM) [6] style approach is a common technique used to select appropriate metrics in empirical studies. This approach guides researchers to: (i) define the goal of measuring maintainability, then (ii) derive external attributes that are possible indicators of maintainability, then (iii) derive from these a set of internal measurable attributes, and finally (iv) derive a set of metrics to measure the internal measurable attributes. Unfortunately, using GQM still leaves a large degree of interpretation to its users, who might independently reach divergent conclusions. One further problem with this uncertainty is that the metric selection process can become circular, especially when measuring maintainability, as external quality attributes are interconnected. For instance, stability indicates maintainability, yet maintainability can be seen as an indicator of stability.

Similar techniques for selecting appropriate metrics in empirical studies have been used in [33]. This study decided to measure attributes such as maintainability, reusability and reliability as indicators of maintainability. From this list, internal attributes such as separation of concerns, coupling, complexity, cohesion and size were selected. The final set of selected coupling metrics was then defined based upon these internal attributes. We can therefore see that uncertainty on key external attributes has great impact on the remainder of the metric selection process.

This lack of conformity on these attributes has unsurprisingly affected the selected coupling metrics. For instance, maintainability is measured in studies[7,15,17] through the application of 9 metrics to measure size, coupling, cohesion and separation of concerns metrics. In[10,33] complexity is in addition derived as an external attribute contributing to maintainability. We return to this topic in Section 5.

Similar problems have been observed in maintainability studies of object-oriented programming (OOP) this has been highlighted in a survey of existing OO empirical studies and their methodologies to predict external quality attributes[5].

Many studies acknowledge that modularity, coupling, cohesion and complexity are internal attributes that affect maintainability. Interestingly, error-proneness was the attribute that was not explicitly derived as an indicator of maintainability.

In short, different interpretations of maintainability and its subsequent derived attributes influence the coupling metrics chosen or defined within the context of an empirical study. This may explain the wide range of coupling metrics observed in AOP empirical studies, which we review in the next subsections.

### 4.2 Coupling Metrics Used to Measure Maintainability

We identified 27 coupling metrics in our sample set of studies. A representative subset of these metrics is shown in Table 3. For each metric, the table lists it's name, description, and six characteristics.

Generally, the most frequent metrics were adapted from object orientation (OO). Among them, the most common were Coupling Between Components (CBC) and Depth of Inheritance Tree (DIT), appearing in 66% of the studies. Adapted metrics hold the advantage of being based upon OO metrics that are widely used, and can be assumed reliable. The (implicit) reasoning is that adapting OO metrics to AOP

maintains their usefulness. This however might no hold: DIT for instance combines both the implicit AO inheritance with the traditional OO inheritance. It thus considers two very different coupling sources together. These sources may have different affects upon maintainability and it may be beneficial to consider these seperately.

In contrast, some of the studies also use coupling metrics developed for AOP, such as Coupling on Advice Execution (CAE) and Number of degree Diffusion Pointcuts (dPC). These metrics enable a more in-depth analysis of the system coupling behaviour, as they consider finer-grained langauge constructs. However, they are more likely to behave unexpectedly, being underdeveloped.

No AO coupling metrics were found to be interchangeable, i.e. none were found to be applicable to different AO languages without any ambiguity. This is probably due to the heterogeneity of AO programming abstractions and mechanisms that makes it very hard to define metrics accurately across multiple AO languages.

The majority of metrics found in our study assess outgoing coupling connections (indicated as "Fan Out" in Table 3). This can be seen as a weakness, as both incoming and outgoing coupling connections help refactoring decisions, as discussed in[31].

**Table 3.** Properties of used coupling metrics

| Metric | Description |
|---|---|
| (DIT) Depth of Inheritance Tree [10] | Longest path from class / aspect to hierarchial root. |
| (RFM) Response for a Module [10] | Methods and advices potentially executed in response to a message received by a given module. |
| (NOC) No. of Children [10] | Immediate sub-classes / aspects of a module. |
| (CBC) Coupling Between Components [10] | Number of classes / aspects to which a class / aspect is coupled. |
| (CAE) Coupling on Advice Execution [10] | No. of aspects containing advice possibly triggered by execution of operations in a given module. |
| (dPC) No. of Degree Diffusion Pointcuts [31] | No. of modules depending on pointcuts defined in the module. |
| (InC) No. of In-Different Concerns [31] | No. of different concerns to which a module is participating. |

| Metric | Measurement Granularity | Measurement Entity | Measurement Type | Fan In / Fan Out | Inter-changeable | AO / Adapted |
|---|---|---|---|---|---|---|
| (DIT) | class / aspect | class / aspect | inheritance | n/a | no | adapted |
| (RFM) | module | method / advice | environmental | fan out | no | adapted |
| (NOC) | module | class / aspect | inheritance | n/a | no | adapted |
| (CBC) | class / aspect | class / aspect | environmental | fan out | no | adapted |
| (CAE) | module | aspect | environmental | fan out | no | AO |
| (dPC) | module | module | environmental | fan in | no | AO |
| (InC) | module | concern | environmental | n/a | no | AO |

### 4.3 Measured AOP Mechanisms

OO coupling metrics can be adapted to take into account AO mechanisms, producing a seemingly equivalent measure. However, this approach might miss some of specific needs of AO programs. We now review how the mechanisms of the AOP languages most commonly used in maintainability studies of AOP were accounted for in coupling measures, and draw attention to mechanisms that are frequently overlooked.

Table 4 lists the mechanisms and abstractions used in the coupling metrics of our study. One first challenge arises from the ambiguity of many notions. For instance, seven metrics use "modules" as their level of granularity, but what is module might vary across languages. In AspectJ an aspect may be considered a module – containing advice, pointcuts and intertype declarations, yet in CaesarJ, each advice forms its own module. More generally, many coupling metrics use ambiguously terms ("module", "concern", or "component") which might be mapped to widely varying constructs in different languages. This hampers the ability of the metrics to draw cross-language comparisons[20].

**Table 4.** AO mechanisms and abstractions accounted for in used coupling metrics

| Abstraction / Mechanism | No. of Metrics | Measurement Entity | Measurement Granularity | Singular Entity Metric |
|---|---|---|---|---|
| Module | 15 | 8 | 12 | 5 |
| Component | 1 | 0 | 0 | 0 |
| Concern | 7 | 5 | 7 | 2 |
| Pointcut | 3 | 0 | 0 | 0 |
| Joinpoint | 2 | 2 | 0 | 2 |
| Intertype Declaration | 1 | 1 | 0 | 0 |
| Aspect | 7 | 4 | 4 | 1 |
| Advice | 3 | 1 | 0 | 0 |

Another challenge comes from the fact that certain phenomenon are best analysed by looking at the base and aspect codes separately. For instance, as a program evolves, it may lose its original structure. However, in AO programs, the base level and aspect level often evolve independently and have different structures. Understanding how each evolution impacts structure thus requires that each be investigated separately. This is not done in most of the empirical studies we found.

We also noted that the majority of used AO metric suites did not focus on interface complexity. This is a problem as AO systems are at risk of creating complex interfaces by extracting code which is heavily dependent on the surrounding base code, and metrics are needed to identify problematic situations[33].

More generally, few studies look at the connection between maintainability and specific AO mechanisms. For instance Response for a Module (RFM) measures connections from a module to methods / advices. This is useful in analysing coupling on a "per module" basis, but does not distinguish between individual AO language constructs. For instance, it adds up intertype declarations jointly with advice as they

both provide functionality that insert extra code into the normal execution flow of the system. However intertype declarations differ from other types of advice as they inject new members (e.g. attributes) into the base code. Coupling metrics have been proposed to address this problem and measure singular mechanisms, such as advice, pointcuts, joinpoints and some intertype declarations[10,26,36], but have rarely been used in maintainability studies.

To sum up, no study used metrics to measure constructs unique to AO programming languages, and very few measured finer-grained language constructs. Although this depends on the particular goals of each maintainability study, this is generally problematic as each mechanism within a particular language has the potential to affect maintainability differently, and should therefore be analysed in its own right.

### 4.4 Validation of Coupling Metrics

Metrics are useful indicators only if they have been validated. There are two complementary approaches to validate software metrics, *empirical validation* and *theoretical validation*[30]. We will focus on the latter. In our context, theoretical validation tests that a coupling metric is accurately measuring coupling and there is evidence that the metric can be an indirect measure of maintainability.

Here we consider the 8 validity properties suggested by Kitchenham[30]. The theoretical criteria are split into two categories: *(i)* properties to be addressed by all metrics; and *(ii)* properties to be satisfied by metrics used as indirect measures. [3] has already used the first criteria on coupling metrics for AO programs. We offer some alternative viewpoints here, and also evaluate the coupling metrics against properties that indirect measures should possess. When we applied this framework to the 27 coupling metrics found in our review, we identified three potential violations of these criteria, discussed below.

*A valid measure must obey the 'Representation Condition'*. This criterion states that there should be a homomorphism between the numerical relation system and the measureable entities. In other words a coupling metric should accurately express the relationship between the parts of the system that it claims to measure. It also implies that coupling metrics should be intuitive of our understanding of program coupling[30]. For instance, a program with a CBC value of 6 should be more coupled than a program with a CBC value of 5. This metric holds true to it's definition, however if a study is using CBC as a representation of coupling within a system this validation criteria becomes questionable. When measuring coupling we often do not perceive each connection as equal. There are different types and strengths of coupling. If we consider two AO systems; the first with 5 coupling connections via intertype declarations, and the second with 5 coupling connections via advice. Even though both systems contain 5 coupling connections, they are not equivalent, and are not equally interdependent. Various sources and types of coupling may influence the interdependency of a system in multiple ways. We found no metrics in the studies that took this finer differences into account.

*Each unit of an attribute contributing to a valid measure is equivalent*. We are assuming that units (modules) that are measured alongside each other are equivalent.

There are some AO coupling metrics that only consider coupling from one language 'unit'. For example, the CAE metric satisfies this property as each connection counted by metric value involves an advice method. However, many metrics used in empirical studies of AOP assume that counting coupling connections between AO modules is equivalent to coupling connections between OO modules. As mentioned b, classes and aspects are often measured together as equivalent modules (e.g in DIT), yet we do not have evidence that they have the same effect upon maintainability, thus violating this criteria.

*There should be an underlying model to justify its construction*. To give good reason for the creation of coupling metrics, there should be underlying evidence that the metric will be an effective indicator of maintainability. Unfortunately, this criterion definition is somewhat circular in the case of maintainability; metrics are often already constructed and applied before supporting this underlying theory and justifying their construction. In OOP it is widely accepted that there is a relationship between coupling and external quality attributes. Because AOP and OOP share similarities, we could infer that metrics that measure a specific form of coupling in OOP hold a similar potential when adapted to AOP (such as DIT, CBC). This however needs to be validated. This need is even more acute for metrics specific to AOP (e.g. CAE), as we have less information on how coupling induced by AOP-specific mechanisms correlate with maintainability.

## 5 Discussion

We first discuss the potential threats to the validity of our study (5.1), and then revisit our original research questions (5.2) in the light of the results we have just discussed.

### 5.1 Threats to Validity

Our study raises both internal and external validity issues. In terms of internal validity, our study is based on 12 papers that matched our criteria (Section 4). This number is not high however this is in line with systematic reviews in software engineering, which often rely on approximately 10 target papers[28]. The size of the sample should however be kept in mind when assessing the generality of our results.

In terms of external validity, we identified a number strengths and liabilities in the state-of-the-art of AO coupling metrics. However, this list is certainly not exhaustive, and does no cover a number of broader issues about AO metrics and maintainability.

For instance, there are certain forms of (semantic) module dependencies that cannot be quantified by conventional coupling metrics, such as those captured by network analysis[40]. The same argument applies to Net Option Values and Design Structure Matrices[9,32]. Finally, AO empirical studies often rely on multiple metrics suites to measure module complexity, module cohesion, and concern properties. Considering coupling in isolation thus limits our horizon, a broader review would be complementary to this work.

### 5.2 Analysis and Implications

The design and use of AO coupling metrics needs to be improved. Analysis of findings revealed problems corresponding to each of the four original research questions.

The selection of metrics to measure maintainability in AO studies is ambiguous. Many issues contribute to this. Some studies specified key external attributes that contribute to the maintainability of a program. The subjectivity and variations of these external attributes(Section 4.1) causes uncertainty of the most effective metrics to select to measure them. Also, deriving attributes that influence maintainability has shown to be a circular process e.g. stability affects maintainability, and maintainability affect stability. Empirical validation may aid researchers to converge on a smaller set of validated coupling metrics. Better-defined metrics will help this process as well (Section 4.4).

Adapted OO metrics are useful to cross-compare AO and OO programs. Naturally, OO coupling metrics that successfully served as valid indicators of maintainability are likely to be re-used. In fact, this assumption applies to many studies that refactor OO programs with aspects. However, it is important that adapted metrics are not the only ones used. Adapted metrics (such as CBC) overlook characteristics that are unique to a particular AOP language as discussed in Section 4.3. For instance, this metric cannot be used to pinpoint the coupling caused by particular AOP constructs, yet specific AOP constructs may impact unexpectedly upon the maintainability of a program. Also, coupling introduced by unique AOP constructs should be also measured as single entities. Otherwise, we are unable to gain in-depth knowledge about the impact of AOP on maintainability.

Some AO metrics provide initial means to measure coupling introduced by specific AO language constructs[10,33,39]. These fine-grained metrics make it easier to locate program elements that are responsible for positive (or negative) results. For example, if we can correlate a high CAE coupling value with poor maintainability, we may infer that specific advice types in AOP languages are harmful.

Also, results from fine-grained AO coupling metrics may facilitate the identification of solutions for classical problems in AOP, such as *pointcut fragility* [26]. Pointcut fragility is the phenomenon associated with instabilities observed in poincut specifications in the presence of changes. It is commonly assumed the syntax-based nature of most pointcut designators is the cause of their fragilities [26]. There are speculations that certain pointcut designators, such as cflow (Section 2.1), cause more instability. These hypotheses re-enforce the need for metrics that quantify specialised types of coupling links between aspectual code and base code. Such envisaged fine-grained metrics would enable us to better understand the effects of particular AOP mechanisms upon maintainability. We need to know which specific mechanisms are typically the cause of high coupling, and does coupling via different mechanisms have the same impact upon maintainability.

There are other important dimensions of coupling beyond granularity, such as direction, or strength of coupling [4]. We identified that the analysed 27 metrics for AOP do overlook important coupling dimensions. This might be misleading conclusions, as different coupling dimensions may affect maintainability in different ways.

Most AO coupling metrics are created with AspectJ as the target language[10,33,36]. However, alternative languages, such as HyperJ and CaesarJ, support AOP based on different mechanisms (Table 1). Section 4.3 discussed the need for coupling metrics tailored to these unique mechanisms of alternative AOP languages. It is also required to define coupling metrics that are interchangeable across these multiple AOP languages.

Not all coupling metrics meet popular validation criteria (Section4.4). Without theoretical validation there is the risk of using metrics that are providing inaccurate results. Even subtle adaptations to widely accepted OO metrics need to be validated. A recurring point in this review was that certain metric definitions assume different language constructs can be measured together as equivalent entities. For instance, coupling via class inheritance in OO programs might not demonstrate equivalent maintainability effects as aspect inheritance in AO programs. Similar effects might also be overlooked in other forms of module specialisations in AOP, such as intertype declarations. Therefore it might not be appropriate to quantify together heterogeneous specialisation forms in AOP.

Liabilities of AO coupling metrics are not restricted to unsatisfactory theoretical validation. Their empirical validation is also limited, and the statistical relevance of coupling metrics' results is compromised. For example, metrics adapted from OOP remain invalidated within the context of AOP. It would be wrong to assume that such adapted metrics can be similarly interpreted in the context of AO software maintainability.

## 6 Conclusions

Conducting the systematic review has presented valuable insights into current trends on coupling measurement for AOP. This has consequently highlighted the need for fine-grained metrics that consider specific AOP constructs. Existing metrics that are frequently used are therefore in danger of overlooking key contributors to maintainability.

For this reason, there is a niche in current maintainability studies of AOP to use coupling metrics that: (i) take specific language constructs into account, (ii) distinguish between the various dimensions of coupling, and (iii) can be applied unambiguously to a variety of AOP languages.

We have also noticed that the maintainability studies of AOP overly concentrate on static coupling metrics. Dynamic coupling metrics [1] for AOP have not been applied in all the analysed studies. This came as a surprise as many AO composition mechanisms rely on the behavioural program semantics. Also, key maintainability attributes, such as error proneness (Section 4.1), are never explicitly derived as an assessment goal.

Validating new metrics is a non-trivial matter. Kitchenham raised the problem that validating metrics solely with predictive models can be problematic [29]. Without theoretical validation, metrics might not be suitable indirect measures of maintainability. It is important to consider the context that a metric is being applied and whether it is an accurate representation of maintainability in AO systems.

Therefore, even AO metrics adapted from empirically-validated OO metrics, can fail to be theoretically sound predictors of maintainability. In fact, our systematic review found that some AO metrics do not obey the representation condition and other criteria.

However, the above goals are difficult to address in one approach. For instance, defining fine-grained metrics to analyse language specific mechanisms is conflicting with the goal of having course-grained metrics that can be applied across multiple AOP languages. Unfortunately, all these goals are crucial for an in-depth comparison of AOP mechanisms. As part of our future works we aim to undertake empirical studies to explore how the goals we have identified may be reconciled in a unified approach.

## References


1. Arisholm, E., Briand, L., Foyen, A.: Dynamic Coupling Measurement for Object-Oriented Software. IEEE Trans. Soft. Eng. 30(8) (2004) 491-506
2. The AspectJ Prog. Guide, http://eclipse.org/aspectj
3. Bartsch, M., Harrison, R.: An Evaluation of Coupling Measures for AOP. LATE Workshop AOSD (2006)
4. Briand, L., Daly, J., Wüst, J.: A Unified Framework for Coupling Measurement in Object-Oriented Systems. IEEE Trans. Software Eng. 25(1) (1999) 91-121
5. Briand, L., Wüst, J. Empirical Studies of Quality Models in Object-Oriented Systems, Advances in Computers. Academic Press (2002)
6. Basili, V., et al.: GQM Paradigm. Comp. Encyclopedia of Soft. Eng. JW&S 1 (1994) 528-532
7. Cacho, N. et al.: Composing design patterns: a scalability study of aspect-oriented programming. AOSD'06 (2006) 109 – 121
8. CaesarJ homepage, http://caesarj.org
9. Cai, Y., Sullivan, K.J.: Modularity Analysis of Logical Design Models. ASE 21 (2006) 91-102.
10. Ceccato, M., Tonella P.: Measuring the Effects of Software Aspectization. WARE cd-rom (2004)
11. Chidamber, S., Kemerer, C.: A Metrics Suite for OO Design. IEEE Trans. Soft. Eng. 20(6) (1994) 476-493
12. Fenton, N. E., Pfleeger, S. L.: Software Metrics: a Rigorous and Practical Approach. 2nd ed. PWS Publishing Co Boston (1998)
13. Figueiredo, E., et al.: Assessing Aspect-Oriented Artifacts: Towards a Tool-Supported Quantitative Method. ECOOP (2005)
14. Filho, F.C., et al.: Exceptions and aspects: the devil is in the details. FSE 14 (2006) 152-156
15. Filho, F.C., Garcia, A. and Rubira, C.M.F.: A quantitative study on the aspectization of exception handling. In Proc. ECOOP (2005)
16. Garcia, A., et al.: On the modular representation of architectural aspects. EWSA (2006)
17. Garcia, A. et al.: Separation of Concerns in Multi-Agent Systems: An Empirical Study. In Software Engineering for Multi-Agent Systems with Aspects and Patterns. J. Brazilian Comp. Soc. 1(8) (2004) 57-72
18. Garcia, A. et al.: Aspectizing Multi-Agent Systems: From Architecture to Implementation.. In R. Choren, A. Garcia, C. Lucena, & A. Romanovsky (Eds.): Software engineering for multi-agent systems III. LNCS, Vol. 3390. Springer-Verlag, (2004) 121-143

19.Garcia, A. et al.: Modularizing Design Patterns with Aspects: A Quantitative Study. In Proc. AOSD (2005) 3-14.
20.Greenwood, P. et al.: On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study. ECOOP (2007) 176-200
21.Harrison, R., Counsell, S., & Nithi., R.: An Overview of Object-Oriented Design Metrics. In Proc. STEP (1997) 230-234.
22.Hitz M, Montezeri, B.: Measuring Coupling and Cohesion in Object-Oriented Systems. In Proc. Int. Symposium on Applied Corporate Computing (1995)
23.Hyper/J home page, http://www.research.ibm.com/hyperspace/HyperJ.htm
24. IEEE Glossaries, http://www.computer.org/portal/site/seportal/index.jsp
25.JBoss AOP, http://labs.jboss.com/jbossaop
26.Kastner, C., Apel, S., and Batory, D.: Case Study Implementing Features Using AspectJ. In Proc. SPLC (2007) 223-232.
27.Kiczales, G. et al.: Aspect-Oriented Programming. ECOOP (1997) 220-242
28.Kitchenham, B., et al.: Systematic Literature Reviews in Software Engineering – A Systematic Literature Review. Information and Software Technology (2008)
29.Kitchenham, B.: Procedures for Performing Systematic Reviews. Joint Tech. Rep. S.E.G. (2004)
30.Kitchenham, B., Pfleeger, S.L., & Fenton, N.: Towards a Framework for Software Validation Measures. IEEE TSE, 21(12) (1995) 929-944
31.Kulesza, U. et al.: Quantifying the Effects of Aspect-Oriented Programming: A Maintenance Study. In Proc. ICSM (2006) 223-233
32.Lopes, C. V., Bajracharya, S.K.: An analysis of modularity in aspect oriented design. AOSD (2005)15-26.
33.Marchetto, A..: A Concerns-based Metrics Suite for Web Applications. INFOCOMP journal of computer science 4 (3) (2004)
34.Pressman, R.S.: Software Engineering: a Practitioner's Approach. McGraw Hill NY (1987)
35.Sant'Anna, C. et al.: On the Modularity of Software Architectures: A Concern-Driven Measurement Framework. In Proc ECSA (2007)
36.Sant'Anna, C. et al.: On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework. In Proc. SBES (2003) 19-34
37.Sant'Anna, C. et al.: On the Modularity of Software Architectures: A Concern-Driven Measurement Framework. In Proc. ECSA (2008)
38.Spring AOP, http://www.springframework.org
39.Zhao, J.: Measuring Coupling in Aspect-Oriented Systems. Int. Soft. Metrics Symp. (2004)
40.Zimmermann, T., Nagappan, N.: Predicting defects using network analysis on dependency graphs. ICSE (2008) 531-540