# An Empirical Evaluation of Coupling Metrics on Aspect-Oriented Programs

Rachel Burrows*
Informatics Department
Pontifical Catholic University
Rio de Janeiro - Brazil
r.burrows@comp.lancs.ac.uk

Fabiano Cutigi Ferrari
Computer Systems Department
University of São Paulo
São Carlos, SP - Brazil
ferrari@icmc.usp.br

Alessandro Garcia
Informatics Department
Pontifical Catholic University
Rio de Janeiro - Brazil
afgarcia@inf.puc-rio.br

François Taïani
Computing Department
Lancaster University
Lancaster - UK
f.taiani@lancaster.ac.uk

## ABSTRACT

Coupling metrics received increased recognition by object-oriented (OO) software developers when they were found to be indicators of important quality attributes, such as fault-proneness. However, there is no consensus on which coupling metrics are effective quality indicators for emerging development paradigms, such as Aspect-Oriented Programming (AOP). AOP aims to improve software quality by providing significantly different decomposition mechanisms, such as pointcut, advice and intertype declarations. Therefore, it is not obvious if quality indicators for AOP can be derived from direct extensions of classical OO metrics. However, empirical studies of AOP do often rely on classical coupling metrics. Despite the recent adoption of AOP in industrial projects, coupling metrics have been rarely evaluated as useful indicators of fault-proneness in this context. This paper analyses the effectiveness of coupling metrics as indicators of fault-proneness in aspect-oriented (AO) systems. We collected faults from several releases of a real-world AO system. We applied and compared existing metrics for coupling and other internal attributes. We have also considered a novel metric that quantifies specific dependencies in AO software not captured by existing metrics. The results show that coupling metrics, which are not directives of object-oriented metrics, tended to be superior indicators of fault-proneness.

## Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Product Metrics*; D.3.3 [**Programming Languages**]: Language Constructs and Features

## General Terms

Measurement, Experimentation, Languages.

## Keywords

Aspect-Oriented software, metrics, fault-proneness.

*Also affiliated with Lancaster University - UK.


*WETSoM* '10, May 2-8 2010, Cape Town, South Africa


## 1. INTRODUCTION

Aspect-Oriented Programming(AOP) [15] has attracted the attention of software researchers and developers as a solution for improving modularity of crosscutting concerns, such as exception handling, concurrency and caching. AOP supports the development of core system functionalities – the *base code* – separately from concerns that cut across them. Once implemented, both base code and the crosscutting modules – the *aspects* – can be combined together to produce a single executable system. The composition of aspects and classes in the base code is implemented through new programming mechanisms, such as *pointcuts*, *advices* and *intertype declarations*.

While AOP is expected to improve software quality [17], its underlying mechanisms establish new forms of connections between base code and aspects. For instance, an advice implements behaviour similarly to a method. However, it is implicitly triggered when specific events (e.g. a method call or a field access) occur during the software execution. *Pointcuts* are defined inside aspects alongside advice to *quantify* the places in the base code for advice to be inserted into. These places in the base code are called *join points*. An *intertype declaration*, on the other hand, implicitly alters the program structure, e.g. by inserting new members (methods or fields) into the base code.

Therefore, it is questionable if extensions of conventional coupling metrics are good quality indicators for AOP. After almost one decade of research on AOP, we can observe that the first quantitative evaluations of AO software are starting to emerge [6, 7, 11, 12]. These studies often rely on metrics that quantify the level of coupling and other properties as indicators of pivotal quality attributes, such as design stability and robustness. However, there has been no convergence or agreement on which coupling metrics are the most effective indicators of fault-proneness. In fact, validation of software metrics for estimating fault-proneness is particularly important in the AOP context, where testing tools are not sophisticated enough.

Existing metrics for AOP [8, 19, 24] have suffered criticism [4, 5, 21], and they are failing to be used with confidence in empirical studies. For instance, such metrics are critisised for being too coarse-grained and not taking into account subtle dimensions of class-aspect coupling. This

problem is magnified by the lack of empirical validation into the effectiveness of existing metrics, which in turn creates uncertainty for developers when deciding on measurement strategies. More specifically, the ability of coupling metrics to indicate fault-proneness still lacks thorough evaluation. To the best of our knowledge, the impact of coupling on the correctness of programs written in AspectJ [16] has never been deeply investigated, despite AspectJ being the most widely adopted AOP technology.

Concerned with the aforementioned issues, this paper analyses the feasibility and effectiveness of using coupling metrics as indicators of fault-proneness in AO programs. This will hopefully expose strengths and weaknesses of existing metrics and pinpoint new research directions for measurement in AOP. To achieve our goals, we evaluated existing metrics for AO software and compared the results with a novel metric that takes into consideration new coupling relationships between aspects and the base code. Data was collected from several releases of a real-world AO system previously used in a study of fault-proneness of AOP mechanisms [10]. The results show that AO-specific coupling metrics are more closely correlated to fault-proneness than the other metrics. Our results also show empirical evidence that motivates an in-depth investigation of the impact of subtle coupling connections on the correctness of AO systems.

The remainder of this paper is organised as follows: Section 2 describes the metrics we used along the study. Next, Section 3 describes the study setup and data collection procedures. The results are analysed and discussed in Section 4. Section 5 summarises the related research and Section 6 concludes this work.

## 2. THE METRICS

A brief description of all metrics used in this study is presented in Table 1.

**Selection of Existing Metrics for AOP**. We selected a representative set of 10 metrics for the analysis. These existing metrics were defined in Ceccato and Tonella's metrics suite [8]. They are a useful benchmark for our analysis because they include extensions to all classical object-oriented metrics proposed by CK metrics [9]. In addition, most of the other available AOP metrics are also extensions of CK metrics (e.g. [8, 19, 24]).

These metrics have been extensively investigated as fault-proneness indicators in object-oriented software [13, 18, 22, 23]. Not only does this metric suite contain a variety of popular coupling metrics, but also contains a selection of other conventional measures for AOP such as cohesion (e.g. LCO) and size (e.g. WOM). This allows us to broaden our analysis, by comparing the predictive power of coupling measures against other conventional metrics.

**A Novel Coupling Metric for AOP**. However, Ceccato and Tonella's metrics do not cover the quantification of all the possible types of class-aspect dependencies. Therefore, for the purpose of our study goals, we created a new metric that quantifies coupling between the base and aspect code, named *Base-Aspect Coupling* (BAC). We designed this metric to overcome limitations [5] of existing coupling metrics for AOP [8, 19, 24]. To support our analysis, we compare BAC to the other metrics described in Table 1.

Note that some metrics proposed by Ceccato and Tonella [8] – in particular, WOM, CBM, RFM, LCO, DIT, and NOC – are adaptations of CK metrics [9] in order to make them applicable to AO programs. Full definition of metrics can be found in the original works [8, 9].

We developed the BAC metric to measure coupling sourced from the following AOP-specific mechanisms: advice (`before`, `after`, `after returning`, `after throwing`, `around`), intertype declarations, `declare soft` statements and `declare parents` statements. More precisely, the BAC metric counts, for each aspect, the number of times these elements affect the base code through the analysis of join point static shadows and structural modifications. Join point static shadows consist of the sets of implicit calls to advice-methods, i.e. bytecode methods that result from the compilation of advices [14]. Such calls are inserted into the base code during the compilation process. The advice execution itself may only be resolved at runtime, that is, not all joint points included in a joint point static shadow will in fact trigger the advice execution.

There are two main characteristics that can be used to differentiate BAC from conventional coupling metrics. Firstly, it only measures AOP-specific coupling; the most popular metrics [8] tend to be more high-level and measure multiple types of coupling together. For instance, the Coupling Between Modules (CBM) metric measures coupling between classes and aspects, and, classes and other classes hence hardening the task of isolating individual effects of AOP mechanisms on fault-proneness of a system. Some other existing metrics have been developed to measure AOP-specific coupling, for example the Coupling on Advice Execution (CAE). Similarly to BAC, this metric measures coupling between the aspect and the base code however it is not sensitive to coupling frequency. That is, with BAC, each time a join point is matched a connection will be aggregated however this is not true for CAE, which only counts the number of classes and aspects coupled regardless of the frequency. To the best of our knowledge, there is no metric that considers these types of dependencies created by AOP mechanisms while also considering coupling strength.

## 3. STUDY SETUP

This section describes our study configuration in terms of its goals (Section 3.1), the selected target application (Section 3.2), and the data collection procedures (Section 3.3).

### 3.1 Study Goals

The main goal of this study is to evaluate the feasibility and effectiveness of using coupling between the base and aspect code as a fault indicator in AO systems. We defined a new coupling metric named Base-Aspect Coupling (BAC) that overcomes limitations of existing coupling metrics used in AOP empirical studies (Section 2). For the purposes of the evaluation we compare the effectiveness of the BAC metric against other popular metrics used in empirical studies of AOP (see Table 1).

### 3.2 The Target System

Selecting a suitable application for this study was difficult process, mainly because there is a lack of available real-world

**Table 1: Metrics used in the study.**

| Metric | Description |
|---|---|
| WOM (Weighted Operations in Module) | Is the number of operations in a class or aspect. Methods, advices and introductions are counted as an operation. It may take into account the internal complexity of the operations[1]. |
| CFA (Coupling on Field Access) | Is the number of classes, aspects or interfaces declaring fields that are accessed by a given class or aspect. |
| CMC (Coupling on Method Call) | Is the number of classes, aspects or interfaces declaring methods that are possibly called by a given class or aspect. |
| CDA (Crosscutting Degree of an Aspect) | Is the number of classes or aspect affected by the pointcuts and by the introductions in a given aspect. |
| CAE (Coupling on Advice Execution) | Is the number of aspects containing advices possibly triggered by the execution of operations in a given class or aspect. |
| RFM (Response For a Module) | Is the number of methods and advices potentially executed in response to a message received by a given class or aspect. |
| LCO (Lack of Cohesion in Operations) | Is the number of pairs of operations working on different class or aspect fields minus pairs of operations working on common fields. |
| CBM (Coupling between Modules) | Is the number of classes, aspects or interfaces declaring methods or fields that are possibly called or accessed by a given class or aspect. |
| DIT (Depth of Inheritance Tree) | Is the length of the longest path from a given class or aspect to the class or aspect hierarchy root. |
| NOC (Number Of Children) | Is the number of immediate sub-classes and sub-aspects of a given module. |
| BAC (Base-Aspect Coupling)[2] | Is the number of join points shadowed from an aspect via advice plus the number of module hierarchy changes from an aspect via intertype declarations, `declare soft` statements and `declare parents` statements. |

[1] Feature not implemented in the AOPMetrics tool [1].

[2] Novel metric investigated in this paper.

AOP-based projects but also because our analysis required accurate fault documentation. On the other hand, based on a previous experience [10], we have observed that the faults followed similar trends in two other systems.

In this study we evaluated AO versions of a Java-based open source framework for object-relational data mapping called iBATIS [2]. iBATIS is typically embedded in Java applications to provide integration between business objects and data persistence. It was originally developed in 2002 and over 60 releases are available at SourceForge.net[1] and Apache.org[2] repositories.

Four releases of iBATIS (builds #150, #174, #203 and #243) were extracted from the SourceForge.net repository. For each release, a subset of concerns were classified as crosscutting and refactored out to aspects, having the OO counterparts as baselines for implementation alignment. This procedure resulted in the four AO releases evaluated in this paper. Each release includes 46 aspects out of 264 modules on average, implemented in ∼11 KLOC. The iBATIS AO releases were first evaluated in a study that focused on fault-proneness of AOP mechanisms [10]. In this paper we use the same fault-related data to evaluate the ability of metrics in predicting faults in AO applications.

## 3.3 Data Collection

The iBATIS system experienced two testing phases in which faults were documented: pre-release and post-release testing. The former was performed by developers of AO releases and aimed to produce defect-free code to be committed to a CVS repository during the aspectisation process. To do so, developers were provided with test sets originally designed for the OO releases. In the post-release phase, we assessed the implementation through the enhancement of the original test sets. In both phases, any fault that could be directly associated with AO mechanisms (i.e. introduced during the aspectisation) was readily documented in an appropriate detailed report. In total, we reported 44 faults discovered at the pre-release phase and 28 faults discovered during post-release tests.

The BAC metric was collected manually using the Cross Reference View provided by the AJDT Eclipse plugin[3]. Rigorous collection strategies were followed during this data collection phase ensuring that all manually collected data was reviewed by two people. All remaining metrics were collected using the AOPMetrics tool [1]. Each module was assigned a fault count according to the total number of faults that was sourced there over all releases. Each module was subsequently given a fault-proneness classification depending on whether a module had a high, medium or low number of faults. "2" was assigned to modules in the 4th (highest) quartile of the fault data set , "1" was assigned to modules in the 3rd quartile and "0" was assigned to modules in the 2nd and 1st quartile. We found no need for separate categories for the lower two quartiles as, due to the distribution of the data over half the modules had 0 faults. Each coupling metric was also classified in the same way.

To analyse the effectiveness of each coupling metric as an indicator of fault-proneness we conducted a Spearman's Rank correlation test. It is a non-parametric test that allows us to measure the correlation between our independent variables (each coupling metric), and our dependent

[1] *http://sourceforge.net/projects/ibatisdb/files/* - 19/02/10

[2] *http://archive.apache.org/dist/ibatis/binaries/ibatis.java* - 19/02/10

[3] *http://www.eclipse.org/ajdt/* - 19/02/2010

variable (faults per module). For this test we used a tool named R language and environment[4]. In order to evaluate the strength of correlation we interpret results with a confidence level of around 95% .

# 4. ANALYSIS AND DISCUSSION

Tables 2 and 3 show the Spearman's rank correlation coefficients of each metric against the number of faults (Table 2), and the fault density (Table 3) of each module. Due to the nature of Crosscutting Degree of an Aspect (CDA) and Base-Aspect Coupling (BAC) their coefficients are defined for aspects only and do not take into account base modules, i.e. the classes and aspects that expose the join points for a given aspect.

**Table 2: Spearman's Rank (# of faults)**

| | Metric | p-value | coefficient |
|---|---|---|---|
| Group A | Crosscutting Degree of an Aspect (CDA) | 0.00000017 | **0.30562110** |
| | Base-Aspect Coupling (BAC) | 0.04646000 | **0.26968580** |
| Group B | Coupling on Method Call (CMC) | 0.01826000 | **0.14072730** |
| | Coupling on Field Access (CFA) | 0.03050000 | **0.12909780** |
| | Coupling between Modules (CBM) | 0.06679000 | **0.10308800** |
| | Weighted Operations in Module (WOM) | 0.10290000 | 0.09748695 |
| Group C | Lack of Cohesion in Operations (LCO) | 0.55900000 | 0.03500226 |
| | Response For a Module (RFM) | 0.99436996 | -0.00044745 |
| | Coupling on Advice Execution (CAE) | 0.70200000 | -0.02157442 |
| | Depth of Inheritance Tree (DIT) | 0.49480000 | -0.03848148 |
| | Number Of Children (NOC) | 0.23540000 | -0.07101578 |

**Table 3: Spearman's Rank (fault densities)**

| | Metric | p-value | coefficient |
|---|---|---|---|
| Group A | Crosscutting Degree of an Aspect (CDA) | 0.04689000 | **0.26918280** |
| | Base-Aspect Coupling (BAC) | 0.04743000 | **0.26854670** |
| Group B | Coupling between Modules (CBM) | 0.01626000 | **0.14324630** |
| | Coupling on Method Call (CMC) | 0.02637000 | **0.13248460** |
| | Coupling on Field Access (CFA) | 0.04092000 | **0.12204390** |
| | Weighted Operations in Module (WOM) | 0.12980000 | 0.09058615 |
| Group C | Lack of Cohesion in Operations (LCO) | 0.60670000 | 0.03083686 |
| | Response For a Module (RFM) | 0.94720000 | -0.00396879 |
| | Coupling on Advice Execution (CAE) | 0.23120000 | -0.07165691 |
| | Number Of Children (NOC) | 0.21090000 | -0.07485521 |
| | Depth of Inheritance Tree (DIT) | 0.07766000 | -0.10543380 |

**Coupling, Faults and their Densities**. We also used fault densities (Table 3) in order to remove the influence of module size on the results, and allow us to gauge the influence of size on the correlation. More precisely, one could argue that larger modules have a higher chance of both: (i) containing higher numbers of faults, and (ii) obtaining higher coupling values, leading to potential correlations in Table 2 that might be better explained by a simple size metric. However, from comparing Tables 2 and 3, size does not seem to impact significantly on correlations, as both tables show very similar results. The ordering of metrics within each group is similar in both tables. The exceptions are CBM, which obtains a better correlation coefficient with fault density (0.14 instead 0.10), and CDA, whose coefficient becomes almost identical to that of Base-Aspect Coupling with fault densities.

The two tables show that the analysed metrics fall in three distinct groups. In Group A, Crosscutting Degree of an Aspect (CDA) and Base-Aspect Coupling (BAC) show the highest levels of correlation. The performance of these metrics is clearly demarcated from the second group, Group B (CMC, CFA, CBM and WOM), which shows much lower levels of correlations. Finally, the third group, Group C (LCO, RFM, CAE, NOC, DIT), contains metrics whose p-value are particularly high, or whose coefficient are negative. A high p-value denotes a low level of confidence in the measured coefficient against a null-hypothesis that the coefficient is null. This fact can be interpreted as an absence of correlation.

**Group A: Quantification Degree as an Effective Indicator?** We can see in Table 2 that the CDA and BAC metrics showed the strongest correlations with coefficient of $\sim$0.31 and $\sim$0.27, respectively. Similar results were obtained when fault densities are considered (Table 3). Both metrics share the common characteristic that they directly or indirectly quantify the degree of join point quantification. Another interesting observation is that the Group A metrics are more closely correlated with faults than other commonly used indicators in empirical studies of AOP, such as cohesion (LCO), size metrics (WOM), and popular coupling metrics (CBM, DIT).

While the CDA metric measures aspect behaviour based only on the pointcut-advice mechanism and introductions, the BAC metric embraces all the other AOP mechanisms such as `declare soft` and `declare parents` statements. The fact that CDA presented a superior correlation might indicate that dependencies created via pointcuts and advice cause more faults than the other dependencies. Figure 1 shows the coupling value from these two metrics for all aspects that contained faults. However, the fluctuations from one module to another are different, this is an indication that they are measuring different sources of fault-proneness. Perhaps, combining these metrics would create a metric that is more closely correlated with fault-proneness.

**Group B: Are Fine-Grained Coupling Metrics Better Indicators?** Earlier research on fine-grained metrics for AOP [5, 20, 21] has suggested that they might be more effective as indicators of certain external quality attributes, such as maintainability or fault-proneness. This assumption might hold for specific cases in the context of fault-proneness. For instance, Coupling on Method Call (CMC) and Coupling on Field Access (CFA) quantify specific dependencies that are both considered by the Coupling Between Modules (CBM) metric. We found that the finer-grained metrics show closer correlations with faults than CBM (Table 2). However, Table 3 does not confirm the superiority of CMC and CFA when fault density is consid-

[4] *http://www.r-project.org/* - 06/02/10

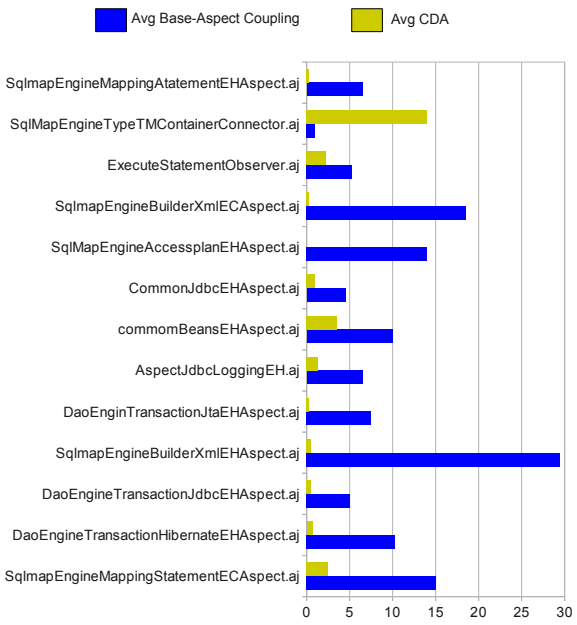


**Figure 1: Average BAC and CDA in faulty modules.**

ered. This observation suggests that, in spite of the fact that fine-grained coupling metrics are expected to be better indicators of faults, our study did not attested this and more research is needed to investigate this issue further.

**Group C: Coupling Granularity and CAE Ineffectiveness**. We also observed that careful attention needs to be paid to other alternative dimensions of coupling as they can drastically influence fault-proneness estimation. For instance, both Base-Aspect Coupling (BAC) and Coupling on Advice Execution (CAE) measure coupling between the base code and the aspects. However, while BAC appeared in the top-two metrics with the closest correlation with faults, CAE showed no correlation. One main difference between these two metrics is the granularity of the measurement entity. On the one hand, CAE is sensitive to the number of aspects that are advising one particular module. On the other hand, BAC is sensitive to the number of times the base modules are being advised by one aspect. A thorough investigation into other coupling dimensions including granularity is therefore needed in the AOP community.

**Coupling, Faults and Instabilities**. Within empirical studies of AOP, specific measurement goals are also likely to affect the coupling metrics that are selected. For instance, code stability is commonly used as a surrogate measure for fault-proneness [11, 12]. Our results show that different coupling metrics are more suitable for measuring stability than those for fault-proneness. We found that CDA showed the closest correlation with faults in our study, however experiments that measured correlations between CDA and code stability found no significant correlation. For instance Bartsch [3] performed an empirical study to test the correlation between 25 coupling metrics for AOP and maintenance effort - the CDA metric showed no correlation. This indicates that an aspect that is highly crosscutting is likely to be fault-prone but not necessarily unstable. It is important that empirical studies of AOP directly evaluate coupling metrics against multiple quality attributes to gain a thorough understanding on the effectiveness of new metrics.

# 5. RELATED WORK

Previous research on coupling metrics for AOP mostly focused on metrics definition and evaluation of software properties such as design stability [11, 12] and robustness [6, 7]. To date, however, there is limited empirical knowledge of the relationship between metrics the fault-proneness is AO software.

Ceccato and Tonella [8] adapted the OO-specific CK metrics [9] to allow them to be applied to AO programs. Besides, they proposed new metrics for measuring coupling that results specifically from AO mechanisms (e.g. CDA – Crosscutting Degree of an Aspect – and CAE – Coupling on Advice Execution) (see Section 2 for more details). However, no empirical empirical evaluation was conducted.

Shen and Zhao [21] empirically evaluated Ceccato and Tonella's metrics. The evaluation also included some new metrics proposed by Shen and Zhao, for example the Crosscutting Degree of an Aspect caused by Pointcuts (CDP), Crosscutting Degree of an Aspect caused by Intertype-declarations (CDI), and Coupling on Advice Execution caused by Advices (CAA). However, differently from our study, they did not correlate the proposed metrics with fault-proneness of the modules, but with other attributes of the programs (such as LOCC - Lines of Class Code). More recently, Shen et al. [20] extended their prior work in a study that evaluates their metrics when applied to check the maintainability of evolving AO programs. Again, the target of the study was different from ours: while they evaluated the relation of a set of metrics with the maintainability of AO programs, we evaluated the relation of a set of metrics with the fault-proneness of AO programs.

Bartsch [3] performed a theoretical and empirical evaluation of 25 coupling metrics for AOP. Similar to our study, he analysed the metrics proposed by Ceccato and Tonella. However, the analysis evaluated the ability of these metrics to be used as indicators of maintenance effort. The best coupling metrics were Response for Module (RFM) and Response for Class (RFC).

# 6. CONCLUSIONS

Our results show that Base-Aspect Coupling (BAC) and Crosscutting Degree of an Aspect (CDA) were the two metrics that displayed the strongest correlation with faults. They outperformed a variety of popular metrics commonly used in AOP empirical studies including cohesion, size and other coupling metrics. Therefore, extensions of CK object-oriented metrics, surprisingly, have not proved to be good indicators of fault-proneness in AOP. Despite a variety of coupling metrics available for AOP, fault-proneness is a complex and multi-faceted property, and it is important not to overlook coupling dimensions sourced from AOP-specific mechanisms. A thorough analysis of finer-grained coupling metrics is imperative to draw accurate conclusions in AOP empirical studies but also for the design of new programming mech-

anisms. In addition, certain dimensions of coupling metrics can have a great impact on their effectiveness. This became clear when we compared Base-Aspect Coupling (BAC) with Coupling on Advice Execution (CAE). Both metrics measured coupling between the base and aspect code. However, one of the best correlations was reported with the former and no correlation was found with the latter.

There is great potential to improve the effectiveness of software measurement in AOP, in particular for the purposes of fault prediction. Note that selecting suitable applications for this kind of study is difficult mainly because there is a lack of available real-world AOP-based projects that include adequate fault documentation. Therefore, in spite of the relatively low correlations between metrics and faults in the analysed system, they provide initial evidences that encourage further investigation. In our future research, we wish to explore different dimensions of coupling between the base and aspect code in AO systems. This area of research within the AOP community is somewhat restricted by the aforementioned lack of readily-available AO benchmark systems as well as by the lack of tooling support for upcoming metrics. We also aim to remedy this in future work.

## Acknowledgments

This work was funded by FAPERJ (distinguished scientist grant E-26/102.211/2009), CNPq (productivity grant 305526/2009-0 and Universal Project grant number 483882/2009-7), PUC-Rio (productivity grant), EPSRC PhD scholarship - UK, and FAPESP (grant 05/55403-6).

## References

[1] AOP Metrics. `http://aopmetrics.tigris.org/` - 19/02/10.

[2] iBATIS. `http://ibatis.apache.org/` - 19/02/2010.

[3] M. Bartsch. *Empirical Assessment of Aspect-Oriented Programming and Coupling Measurement in Aspect-Oriented Systems*. PhD thesis, School of Systems Engineering - University of Reading, UK, 2007.

[4] M. Bartsch and R. Harrison. An evaluation of coupling measures for AspectJ. In *LATE'06 Workshop*. ACM Press, 2006.

[5] R. Burrows, A. Garcia, and F. Taïani. Coupling metrics for aspect-oriented programs: A systematic review of maintainability studies. In *ENASE'09*. Springer, 2009.

[6] N. Cacho, F. Castor Filho, A. Garcia, and E. Figueiredo. Ejflow: taming exceptional control flows in aspect-oriented programming. In *AOSD'08*, pages 72–83. ACM Press, 2008.

[7] F. Castor Filho, N. Cacho, E. Figueiredo, R. Maranhão, A. Garcia, and C. M. F. Rubira. Exceptions and aspects: The devil is in the details. In *FSE'06*, pages 152–162. ACM Press, 2006.

[8] M. Ceccato and P. Tonella. Measuring the effects of software aspectization. In *WARE'04 Workshop*, 2004.

[9] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[10] F. C. Ferrari et al. An exploratory study of fault-proneness in evolving aspect-oriented programs. In *ICSE'10*. ACM Press, 2010. (to appear).

[11] E. Figueiredo et al. Evolving software product lines with aspects: An empirical study on design stability. In *ICSE'08*, pages 261–270. ACM Press, 2008.

[12] P. Greenwood et al. On the impact of aspectual decompositions on design stability: An empirical study. In *ECOOP'07*, pages 176–200 (LNCS v.4609), Berlin - Germany, 2007. Springer Berlin.

[13] T. Gyimóthy, R. Ferenc, and I. Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, 31(10):897–910, 2005. ISSN 0098-5589.

[14] E. Hilsdale and J. Hugunin. Advice weaving in AspectJ. In *AOSD'04*, pages 26–35. ACM Press, 2004.

[15] G. Kiczales et al. Aspect-oriented programming. In *ECOOP'97*, pages 220–242 (LNCS v.1241). Springer-Verlag, 1997.

[16] G. Kiczales et al. Getting started with AspectJ. *Communications of the ACM*, 44(10):59–65, 2001.

[17] R. Laddad. Aspect-oriented programming will improve quality. *IEEE Software*, 20(6):90–91, 2003.

[18] H. M. Olague, L. H. Etzkorn, S. Gholston, and S. Quattlebaum. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Transactions on Software Engineering*, 33(6):402–419, 2007. ISSN 0098-5589.

[19] C. Sant'Anna, A. Garcia, C. Chavez, C. Lucena, and A. von Staa. On the reuse and maintenance of aspect-oriented software: An assessment framework. In *17$^{th}$ Brazilian Symposium on Software Engineering (SBES)*, pages 19–34, Manaus - Brazil, 2003.

[20] H. Shen, S. Zhang, and J. Zhao. An empirical study of maintainability in aspect-oriented system evolution using coupling metrics. In *TASE'08*, pages 233–236. IEEE Computer Society, 2008.

[21] H. Shen and J. Zhao. An evaluation of coupling metrics for aspect-oriented software. Technical Report SJTU-CSE-TR-07-04, Center for Software Engineering, SJTU, Shanghai - China, 2007.

[22] R. Subramanyam and M. Krishnan. Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Transactions on Software Engineering*, 29(4):297–310, 2003.

[23] M.-H. Tang, M.-H. Kao, and M.-H. Chen. An empirical study on object-oriented metrics. In *METRICS'99*, pages 242–249. IEEE Computer Society, 1999.

[24] J. Zhao. Measuring coupling in aspect-oriented systems. In *METRICS'04, (Late Breaking Paper)*, Chicago/IL, USA, 2004.