A Systematic Review on Evaluation of Aspect Oriented Programming using Software Metrics

Joyce M. S. França and Michel dos Santos Soares
Computing Faculty, Federal University of Uberlândia, Uberlândia, Brazil

Keywords: Aspect-oriented Programming, Systematic Review, Software Metrics.

Abstract: Aspect oriented software development has been applied in past years with the promise of improving modularization by addressing crosscutting concerns. Many studies have been published, with varying degree of success on using this paradigm. Software metrics have been presented with the purpose of evaluating the final results. However, too few studies on empirical evidence of the benefits of aspect-oriented paradigm were published, results are frequently subjective, and some studies are non-conclusive. A systematic review on aspect oriented software development and metrics is proposed in this article.

1 INTRODUCTION

Object-Oriented Programming (OOP) emerged with the promise of improving modularization and increasing the abstraction level to programming and design. Despite the success of OOP, the benefits are still not enough in the sense that it did not solve issues of modularity. OOP is appropriate at modularizing core concerns, but fails when modularizing crosscutting concerns, i.e., concerns scattered throughout implementation (Filman et al., 2005). For instance, when dealing with behaviours that span many, often unrelated modules, OOP can be inadequate. The negative impact of code tangling and code scattering affects software development in many ways, including poor traceability, low productivity, problems when trying to reuse code, poor overall quality and increased efforts for software maintenance (Laddad, 2003). A new implementation construct, the aspect (Kiczales et al., 1997), was introduced in order to deal with these issues.

Two main problems motivated this study. The first one is considering the fact that too few studies on empirical evidence of the benefits of Aspect-Oriented Programming (AOP) were published. In addition to this first problem, a derived problem is that studies are non-conclusive. For instance, researchers came to conclusions that AOP present many benefits, despite drawbacks such as decreased performance (Mortensen et al., 2012). Other results have shown that AOP does not provide real benefits or the benefits are marginal (Madeyski and Szala, 2007) (Bartsch and Harrison, 2008), and there are yet other publications clearly advising against AOP (Przybylek, 2010) (Przybylek, 2011). Another issue is that benefits are normally shown without explicit measurements or results are subjective (Ali et al., 2010).

2 RESEARCH METHOD

The main research question that motivated this research is: *What evidence currently exist in the literature that aspect-oriented software development is beneficial?* A derived research question is "*how the evidence of benefits of AOP has been measured?*". The answer to this question is related to what metrics have been applied in order to evaluate AOP, and how AOP compares with OOP.

In order to answer both questions, the proposal is to perform a systematic review (Kitchenham, 2004) with the purpose of identifying what type of research has been performed within AOP. The systematic review started with a search in a number of software engineering conferences and journals. The search was performed starting on January 2006 up until January 2012, i.e., the past 6 years. The chosen conferences were: CSMR (European Conference on Software Maintenance and Reengineering), ICSE (International Conference on Software Engineering), AOSD (International Conference on Aspect-Oriented Software Development), ECOOP (European Conference on Object-Oriented Programming), and OOP-

M. S. França J. and dos Santos Soares M..
A Systematic Review on Evaluation of Aspect Oriented Programming using Software Metrics.
DOI: 10.5220/0003985300770083
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 77-83
ISBN: 978-989-8565-11-2

Table 1: Overview of Search Results and selected papers.

Venues	Retrieved Papers	Pre-selected papers	Selected Papers	Search Method	Data source
CSMR	4	3	0	automatic	IEEExplorer
ECOOP	5	2	2	manual	-
ICSE	26	6	2	automatic	IEEExplorer
AOSD	33	9	4	automatic	ACM digital library
OOPSLA	13	0	0	automatic	ACM digital library
JSS	40	5	2	automatic	ScienceDirect
TOSEM	4	1	0	automatic	ACM digital library
TSE	4	1	1	automatic	IEEExplorer
IST	32	6	2	automatic	ScienceDirect
Total	161	33	13		

Table 2: Selected papers.

	Venue/year	Reference
1	AOSD/06	(Cacho et al., 2006)
2	AOSD/08	(Cacho et al., 2008)
3	AOSD/11	(Ramirez et al., 2011)
4	AOSD/10	(Hovsepyan et al., 2010)
5	ECOOP/08	(Coelho et al., 2008)
6	ECOOP/07	(Greenwood et al., 2007)
7	ICSE/08	(Hoffman and Eugster, 2008)
8	ICSE/08	(Figueiredo et al., 2008)
9	IST/08	(Malta and Valente, 2009)
10	IST/10	(Tizzei et al., 2011)
11	TSE/10	(Mortensen et al., 2012)
12	JSS/08	(Kouskouras et al., 2008)
13	JSS/11	(d'Amorim and Borba, 2010)

SLA (International Conference on Object Oriented Programming, Systems, Languages and Application). Papers published on specific workshops held together with these conferences were not considered. The chosen journals were: JSS (Journal of Systems and Software), TOSEM (ACM Transactions on Software Engineering Methodology), TSE IEEE (IEEE Transactions on Software Engineering), and IST (Information and Software Technology).

The chosen search string was ("aspect oriented" AND "metrics"). The total number of retrieved papers was 161. The criteria to include a paper was divided into two steps. In the first step, for each paper, the title, the keywords and the abstract were read in order to make the selection. When in doubt if the paper should be added, also the introduction and/or the conclusion were read. In this first step, articles with focus on AOP as main concern and with some measurements as results were considered. Even though a large number of papers were retrieved, only a small part was fully read (33 papers). The reason is because many "false positives" were retrieved. The word "aspect" is generally used in another context. For the second step, the criteria for consideration was to fully read the 33 papers with focus on the application of software metrics in the first version of at least one software, then the identification of crosscutting concerns followed by software refactoring with aspects, followed by the application of software metrics in the refactored software version and a comparison of metrics for each software version (OOP and AOP). As a result, thirteen papers were chosen for further analysis, as described in table 2. Within these, twenty-nine studies were considered. The other 20 papers were not selected mainly because, although presented AOP issues, their focus was not on software metrics or the evaluation was poor or non-existent.

3 CRITERIA FOR EVALUATION

The selected papers were evaluated based on criteria of type of evaluation, number of studies, properties and metrics, and which crosscutting concerns were implemented as aspects, as described in this section.

3.1 Type of Evaluation

The evaluation of each one of the selected papers was classified in: real industrial environment, controlled experiment, case study, and toy application. In this article, real industrial environment means that the research was applied in practice in a company, when engineering real software products. This is different

Table 3: Summary of studies.

Type of evaluation	Article	Number of studies	Application	Size (LOC)	Original PL	Aspect PL
Industrial environment	(Ramirez et al., 2011)	1	Plato	1212	Java	AspectJ
	(Mortensen et al., 2012)	3	InstanceDrivers	1600	C++	AspectC++
			PowerAnalyzer	13900		
			ErcChecker	51600		
	(Kouskouras et al., 2008)	1	Telecom	1700	Java	AspectJ
Controlled experiment	(Hovsepyan et al., 2010)	2	Toll System	363	Java	AspectJ
			Pacemaker	369		
Case study	(Hovsepyan et al., 2010)	2	Toll System	363	Java	AspectJ
			Pacemaker	369		
	(Cacho et al., 2006)	3	Middleware	NA	Java	AspectJ
			Measurement tool			
			Agent-based			
	(Cacho et al., 2008)	1	MobileMedia	4000	Java ME	AspectJ
	(Coelho et al., 2008)	3	Health Watcher	8825	Java	AspectJ
			Mobile Photo	1571		
			JHotDraw	21027		
	(Greenwood et al., 2007)	1	Health Watcher	4000	Java	AspectJ, CaesarJ
	(Hoffman and Eugster, 2008)	3	Telestrada	3400	Java	AspectJ
			Pet Store J2EE	17800		
			Health Watcher	4000		
	(Figueiredo et al., 2008)	2	MobileMedia	3000	Java	AspectJ
			BestLab	10000		
	(Malta and Valente, 2009)	4	Jaccounting	11676	Java	AspectJ
			JHotDraw	40022		
			Prevayler	2418		
			Tomcat	45107		
	(Tizzei et al., 2011)	1	MobileMedia	11000	Java ME	AspectJ
	(d'Amorim and Borba, 2010)	1	Health Watcher	5500	Java	AspectJ
Toy App	(d'Amorim and Borba, 2010)	1	Library System	600	Java	AspectJ

from a case study, which involves working with some sample application, often an open-source project. A controlled experiment means that an experimental approach for evaluation was followed with human subjects participating and being evaluated when executing tasks. A toy application is normally a software tool developed in academia in order to prove research concepts.

Other fields of comparison include the total number of studies presented in each article, the name of the application(s) developed and presented in the article, the size of the application given in LOC (NA means not available), the original programming language in which the application was developed, and the AOP language used in the refactoring process. Some of the selected articles present more than one application as study. Therefore, Table 3 presents the number of studies in one column. When the article had more than one study, the results of each metric are compared for each study. For example, if the article has two studies, and the result of the metric was positive/negative for both, for this metric the article obtain positive/negative conclusion. However, if one study has positive result and the other one negative result, then this article receive inconclusive mark.

3.2 Considered Crosscutting Concerns

The list of the most common crosscutting concerns considered (Filman et al., 2005) 4 and the acronyms used in this article are: S - Security, P - Persistence, C - Caching, T - Tracing, NPC - Null Pointer Checking, EH - Exception Handling, RTC - Run-Time Configurations, Sy - Synchronization, CC - Concurrency Control, TM - Transaction Management, CIC - Class Initialization Checking, DS - Domain Specific.

Table 4: Crosscutting Concerns implemented in the studies.

Articles	Types of Crosscuting Concerns transformed in aspects											
	S	P	C	T	NPC	EH	RTC	Sy	CC	TM	CIC	DS
(Cacho et al., 2006)												
(Cacho et al., 2008)						●						
(Ramirez et al., 2011)												
(Hovsepyan et al., 2010)	●							●				●
(Coelho et al., 2008)		●				●			●	●		●
(Greenwood et al., 2007)		●				●			●			●
(Hoffman and Eugster, 2008)						●			●			
(Figueiredo et al., 2008)		●				●						
(Malta and Valente, 2009)						●					●	●
(Tizzei et al., 2011)						●						
(Mortensen et al., 2012)			●	●	●	●	●					●
(Kouskouras et al., 2008)												●
(d'Amorim and Borba, 2010)		●							●			●
Total	1	4	1	1	1	8	1	1	4	1	1	7

Table 5: Properties and Metrics - 1.

Property	Metric	Articles	Conclusion	Total
Code Size	Lines of Code (LOC)	(Greenwood et al., 2007)	+	2+, 7-, 3?
		(Mortensen et al., 2012)	+	
		(Cacho et al., 2006)	-	
		(Cacho et al., 2008)	-	
		(Ramirez et al., 2011)	-	
		(Coelho et al., 2008)	-	
		(Figueiredo et al., 2008)	-	
		(Malta and Valente, 2009)	-	
		(Tizzei et al., 2011)	-	
		(Hovsepyan et al., 2010)	?	
		(d'Amorim and Borba, 2010)	?	
		(Hoffman and Eugster, 2008)	?	
	Vocabulary Size (VS)	(Cacho et al., 2006)	+	1+, 6-
		(Cacho et al., 2008)	-	
		(Greenwood et al., 2007)	-	
		(Figueiredo et al., 2008)	-	
		(d'Amorim and Borba, 2010)	-	
		(Coelho et al., 2008)	-	
		(Tizzei et al., 2011)	-	
	Number of attributes (NOA)	(Cacho et al., 2006)	+	3+, 1?
		(Ramirez et al., 2011)	+	
		(Hovsepyan et al., 2010)	+	
		(Cacho et al., 2008)	?	
	Number of operations (NOO)	(Hoffman and Eugster, 2008)	-	1-
	Weighted operations per component (WOC)	(Cacho et al., 2006)	+	3+, 2-
		(Ramirez et al., 2011)	+	
		(Hovsepyan et al., 2010)	+	
		(Cacho et al., 2008)	-	
		(Greenwood et al., 2007)	-	
Complexity	Ciclomatic Complexity per component	(Ramirez et al., 2011)	+	1+

3.3 Properties and Metrics

Metrics were grouped into properties. Properties such as code size, cohesion and coupling are useful to evaluate the chosen architecture design. Tables 5 and 6 present the final list of properties and metrics. We extracted from each article which metrics were used. Moreover, we came to this conclusion obtained from each article about comparing AOP applications and non-AOP applications. The conclusion could be positive (+), negative (-) or inconclusive (?). Inconclusive results are based on two possibilities. Either metrics were applied in two different applications and the results are nearly the same, or one application obtained positive conclusions and the other one negative conclusions.

Table 6: Properties and Metrics - 2.

Property	Metric	Articles	Conclusion	Total
Coupling	Coupling between Components (CBC)	(Hovsepyan et al., 2010)	+	3+, 3-, 1?
		(Greenwood et al., 2007)	+	
		(Figueiredo et al., 2008)	+	
		(Cacho et al., 2006)	-	
		(Cacho et al., 2008)	-	
		(d'Amorim and Borba, 2010)	-	
		(Hoffman and Eugster, 2008)	?	
	Depth of Inheritance Tree (DIT)	(Ramirez et al., 2011)	+	1+, 2-
		(Cacho et al., 2008)	-	
		(d'Amorim and Borba, 2010)	-	
	Efferent Coupling (EC)	(Ramirez et al., 2011)	+	1+, 2-
		(Tizzei et al., 2011)	-	
		(Kouskouras et al., 2008)	-	
Cohesion	Lack of Cohesion over Operations (LCOO)	(Cacho et al., 2006)	+	6+, 3-
		(Ramirez et al., 2011)	+	
		(Hovsepyan et al., 2010)	+	
		(Greenwood et al., 2007)	+	
		(Tizzei et al., 2011)	+	
		(d'Amorim and Borba, 2010)	+	
		(Cacho et al., 2008)	-	
		(Hoffman and Eugster, 2008)	-	
		(Figueiredo et al., 2008)	-	
Separation of Concerns	Concern diffusion over components(CDC)	(Hovsepyan et al., 2010)	+	3+, 3-
		(Tizzei et al., 2011)	+	
		(d'Amorim and Borba, 2010)	+	
		(Cacho et al., 2008)	-	
		(Greenwood et al., 2007)	-	
		(Figueiredo et al., 2008)	-	
	Concern diffusion over operations (CDO)	(Hovsepyan et al., 2010)	+	3+, 2-
		(Greenwood et al., 2007)	+	
		(Tizzei et al., 2011)	+	
		(Cacho et al., 2008)	-	
		(Figueiredo et al., 2008)	-	
	Concern diffusion over LOC (CDLOC)	(Cacho et al., 2008)	+	6+
		(Hovsepyan et al., 2010)	+	
		(Greenwood et al., 2007)	+	
		(Figueiredo et al., 2008)	+	
		(Tizzei et al., 2011)	+	
		(Mortensen et al., 2012)	+	
Change Impact	Number of added/changed/removed components	(d'Amorim and Borba, 2010)	+	1+, 2-, 2?
		(Tizzei et al., 2011)	-	
		(Tizzei et al., 2011)	-	
		(Greenwood et al., 2007)	?	
		(Figueiredo et al., 2008)	?	
	Number of added/changed/removed operations	(Greenwood et al., 2007)	+	2+, 1-
		(Figueiredo et al., 2008)	+	
		(Tizzei et al., 2011)	-	

4 DISCUSSION

Even after performing the systematic research, the first research question: *What evidence currently exist in the literature that aspect-oriented software development is beneficial?* still cannot be properly answered. Most works implement less than three crosscutting concerns as aspects, which means that too many crosscutting concerns were not considered. For instance, considering that 26 applications were performed within these 13 articles, the important concern of logging was not implemented. The research question: "*how the evidence of benefits of AOP has been measured?*", has been answered mainly with the application of metrics proposed for object-oriented programming. However, many important metrics, such

as “Response for a component” and “Operation cohesion”, which are related to the Modularity property, were not considered at all. Another issue is that few specific aspect-oriented metrics were applied in the studies.

In general, it was difficult to conclude the benefits of AOP in terms of the code size property. Seven studies presented negative results (the number of lines of code increased with AOP), two positives, and three inconclusive. According to these results, for most studies AOP increased LOC after refactoring with aspects. This result is surprising, as it should be expected that with better modularization with aspects, and the deletion of lines of code corresponding to crosscutting concerns, the number of LOC should decrease. The metric vocabulary size had negative results for three studies, which means that the number of elements such as classes, interfaces and configuration files increased in the AOP version. This is natural, as with the introduction of aspects further connection between code elements is necessary. The metric “number of attributes” has positive results when refactoring with AOP. As only one article mentioned the metric “number of operations”, it is not possible to draw conclusions. The metric “weighted operations per component (WOC)” was mentioned in five articles, and eight studies, with six positives and two negatives. It seems that AOP decreases WOC, i.e., decreases complexity of a component. However, further analysis is necessary.

With only one article, and only one application, it is not possible to draw conclusions on complexity. Also, all metrics about coupling (CBC, DIT, EC) are inconclusive, which means that it was not possible to state that AOP will have any benefit in terms of decreasing coupling. DIT and EC had one positive, and one negative each. CBC had two negatives, three positives, and one inconclusive result. Cohesion was investigated in fourteen applications. It was improved in eight applications, but negative in six applications. Based on this, although it seems that cohesion is increased with AOP, it is not possible to draw strong conclusions. About the “Separation of Concerns”, three metrics were considered: CDC, CDO, and CDLOC. The first two are inconclusive, but the metric of concern diffusion over LOC was positive for all six articles.

Change impact was defined through two metrics. The first one, “Number of added/changed/removed class (modules)”, is insignificant for two articles, and negative for one article. Based on this, is seems that the application of AOP does not have a strong impact on classes, but further analysis is necessary. The second one, “Number of added/changed/removed operations”, is positive (AOP makes few changes) for two articles, and negative (AOP makes many changes) for one. Based on this, it is difficult to come to strong conclusions. From Table 4, it is clear that most implemented crosscutting concern was “Exception Handling”. Therefore, it can be inferred that aspects are very used to modularize exceptions. Only four out of thirteen papers implemented three or more crosscutting concerns, i.e., most researchers are not introducing aspects in substitution of many crosscutting concerns. The reason for this policy is not clear and should be further investigated.

5 CONCLUSIONS

The results of the systematic review proposed in this article are that too few studies on empirical evidence of the benefits of aspect-oriented paradigm were published, results are frequently subjective, and some studies are non-conclusive. Only a small number of crosscutting concerns are actually being considered by most researches. In addition, too few software metrics were applied, which makes it difficult to draw strong conclusions, and specific AOP metrics were applied in only few studies. In terms of type of evaluation, much more has to be done. Only two studies were held in a real industrial environment, and only one controlled experiment was performed. It is surprising that for most studies, the number of lines of code increased. As the introduction of aspects increase modularity, it could be expected that the number of lines of code would decrease. It is also surprising that software metrics that are important for architectural properties, such as coupling and modularity, were not investigated in any of the searched papers.

Much more can be done on evaluating AOP. One important result is that the introduction of AOP in industry is still in its infancy, as only five out of twenty-nine studies were performed in a real industrial environment. Controlled experiments with developers is also not common (only one article), which means that there is a large path to follow on this matter. Further studies are necessary to understand what happens with complexity with the introduction of aspects in an OO code, as only one article mentioned the cyclomatic complexity metric.

REFERENCES

Ali, M. S., Ali Babar, M., Chen, L., and Stol, K.-J. (2010). A Systematic Review of Comparative Evidence of

Aspect-Oriented Programming. *Information and Software Technology*, 52:871–887.

Bartsch, M. and Harrison, R. (2008). An Exploratory Study of the Effect of Aspect-Oriented Programming on Maintainability. *Software Quality Control*, 16:23–44.

Cacho, N., Filho, F. C., Garcia, A., and Figueiredo, E. (2008). EJFlow: Taming Exceptional Control Flows in Aspect-Oriented Programming. In *Proc. of the 7th International Conference on Aspect-Oriented Software Development*, pages 72–83.

Cacho, N., Sant'Anna, C., Figueiredo, E., Garcia, A., Batista, T., and Lucena, C. (2006). Composing Design Patterns: a Scalability Study of Aspect-Oriented Programming. In *Proc. of the 5th International Conference on Aspect-Oriented Software Development*, pages 109–121.

Coelho, R., Rashid, A., Garcia, A., Ferrari, F. C., Cacho, N., Kulesza, U., von Staa, A., and de Lucena, C. J. P. (2008). Assessing the Impact of Aspects on Exception Flows: An Exploratory Study. In *ECOOP*, pages 207–234.

d'Amorim, F. and Borba, P. (2010). Modularity Analysis of Use Case Implementations. In *Brazilian Symposium on Software Components, Architectures and Reuse*, pages 11–20.

Figueiredo, E., Cacho, N., Sant'Anna, C., Monteiro, M., Kulesza, U., Garcia, A., Soares, S., Ferrari, F., Khan, S., Castor Filho, F., and Dantas, F. (2008). Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. In *Proc. of the 30th international conference on Software engineering*, pages 261–270.

Filman, R. E., Elrad, T., Clarke, S., and Akşit, M., editors (2005). *Aspect-Oriented Software Development*. Addison-Wesley, Boston.

Greenwood, P., Bartolomei, T. T., Figueiredo, E., Dósea, M., Garcia, A. F., Cacho, N., Sant'Anna, C., Soares, S., Borba, P., Kulesza, U., and Rashid, A. (2007). On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study. In *ECOOP*, pages 176–200.

Hoffman, K. and Eugster, P. (2008). Towards Reusable Components with Aspects: An Empirical Study on Modularity and Obliviousness. In *Proc. of the 30th International Conference on Software Engineering*, pages 91–100.

Hovsepyan, A., Scandariato, R., Van Baelen, S., Berbers, Y., and Joosen, W. (2010). From Aspect-Oriented Models to Aspect-Oriented Code?: The Maintenance Perspective. In *Proc. of the 9th International Conference on Aspect-Oriented Software Development*, pages 85–96.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J. (1997). Aspect-Oriented Programming. volume 1241 of *Lecture Notes in Computer Science*, chapter 10, pages 220–242.

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. Keele university. technical report tr/se-0401, Department of Computer Science, Keele University, UK.

Kouskouras, K. G., Chatzigeorgiou, A., and Stephanides, G. (2008). Facilitating Software Extension with Design Patterns and Aspect-Oriented Programming. *Journal of Systems and Software*, 81:1725–1737.

Laddad, R. (2003). *AspectJ in Action*. Manning, USA, 1 edition.

Madeyski, L. and Szala, L. (2007). Impact of Aspect-Oriented Programming on Software Development Efficiency and Design Quality: an Empirical Study. *IET Software*, 1(5):180–187.

Malta, M. N. and Valente, M. T. O. (2009). Object-Oriented Transformations for Extracting Aspects. *Information and Software Technology*, 51(1):138–149.

Mortensen, M., Ghosh, S., and Bieman, J. M. (2012). Aspect-Oriented Refactoring of Legacy Applications: An Evaluation. *IEEE Transactions on Software Engineering*, 38:118–140.

Przybylek, A. (2010). What is Wrong with AOP? In *ICSOFT (2)*, pages 125–130.

Przybylek, A. (2011). Impact of Aspect-Oriented Programming on Software Modularity. In *Proc. of the 15th European Conference on Software Maintenance and Reengineering*, pages 369–372.

Ramirez, A. J., Jensen, A. C., and Cheng, B. H. (2011). An Aspect-Oriented Approach for Implementing Evolutionary Computation Applications. In *Proc. of the Tenth International Conference on Aspect-Oriented Software Development*, pages 153–164.

Tizzei, L. P., Dias, M. O., Rubira, C. M. F., Garcia, A., and Lee, J. (2011). Components Meet Aspects: Assessing Design Stability of a Software Product Line. *Information & Software Technology*, 53(2):121–136.