Aspect-Oriented Software Maintenance Metrics: A Systematic Mapping Study

Juliana Saraiva
Informatics Center, UFPE
jags2@cin.ufpe.br

Emanoel Barreiros
Informatics Center, UFPE
efsb@cin.ufpe.br

Adauto Almeida
Informatics Center, UFPE
ataf@cin.ufpe.br

Flávio Lima
Informatics Center, UFPE
fal2@cin.ufpe.br

Aline Alencar
Informatics Center, UFPE
aaac@cin.ufpe.br

Gustavo Lima
Informatics Center, UFPE
ghlp@cin.ufpe.br

Sergio Soares
Informatics Center, UFPE
scbs@cin.ufpe.br

Fernando Castor
Informatics Center, UFPE
fjclf@cin.ufpe.br

***Abstract*—Background: Despite the number of empirical studies that assess Aspect-Oriented Software Development (AOSD) techniques, more research is required to investigate, for example, how software maintainability is impacted when these techniques are employed. One way to minimize the effort and increase the reliability of results in further research is to systematize empirical studies in Aspect-Oriented Software Maintainability (AOSM). In this context, metrics are useful as indicators to quantify software quality attributes, such as maintenance. Currently, a high number of metrics have been used throughout the literature to measure software maintainability. However, there is no comprehensive catalogue showing which metrics can be used to measure AOSM. Aim: To identify an AOSM metrics suite to be used by researchers in AOSM research. Method: We performed a systematic mapping study based on Kitchenham and Charters' guidelines, which derived a research protocol, and used well known digital libraries engines to search the literature. Conclusions: A total of 138 primary studies were selected. They describe 67 aspect-oriented (AO) maintainability metrics. Also, out of the 575 object-oriented maintainability metrics that we analyzed, 469 can be adapted to AO software. This catalogue provides an objective guide to researchers looking for maintainability metrics to be used as indicators in their quantitative and qualitative assessments. We provide information such as authors, metrics description, and studies that used the metric. Researchers can use this information to decide which metrics are more suited for their studies.**

I. Introduction

Although computer science is an area of knowledge related to exact sciences, Software Engineering (SE) has very peculiar characteristics that strongly relate it to social sciences. Considering this fact and the need for reliable results, it is imperative to encourage the implementation of empirical studies that are able to assess the effectiveness of techniques, methodologies and processes proposed in the area. Experimentation, one kind of empirical study, allows the knowledge to be generated in a systematic, disciplined, quantifiable and controlled way, especially in areas where human interaction is a dominant factor [1]. For this reason, experiments are commonplace in social and behavioral sciences.

On the other hand, these characteristics make experiments very difficult to plan and evaluate, since it is not possible to generate accurate models as in Mathematics or Physics. Even if an experiment is well-planned, there are still many subjective variables that are very difficult to isolate, such as human interactions and behaviors in software development processes. One way to aid organizations and prove evidence about the benefits of new technology is to apply it in some situation that is, at least, close to a real one. However, most companies are not willing to risk a project using a new technology that was not throughly assessed or whose benefits were not yet demonstrated or justified. Empirical studies are essential to fill this gap, providing reliable data about a given technology, easing its transfer and adoption. This issue has been widely discussed by several researchers [1, 2, 3, 4, 5, 6, 7].

Maintainability is considered a software attribute that plays an important role in its quality level. The less effort/cost the software maintenance cycle requires, the higher the software's quality level. Hence, new software development methods, techniques, and tools often aim to minimize future costs in the maintenance process. Aspect-Oriented Software Development (AOSD) is a widely used software technology in academic scenarios [8]. It aims to improve the structure of software systems by promoting the modularization of concerns whose realization is scattered throughout system modules [8]. Its adoption in software companies could increase if more empirical studies were conducted, providing accurate information about its benefits and drawbacks. An approach to reduce the time for a new technology to make the transition from academia to everyday software development is to show when its impact is beneficial or not in software maintainability [9, 4, 10, 11, 12]. Thus, the focus of this work is on Aspect Oriented Software Maintainability (AOSM).

Various metrics suites have been used as indicators in quantitative and qualitative software engineering research, as depicted by the primary studies listed in Appendix A. The literature provides a large number of metrics to analyze different software characteristics in object-oriented and aspect-oriented software development. As a consequence of the number and diversity of existing metrics, it is difficult for researchers to choose the most appropriate set of metrics for an empirical study. In this context, we performed a systematic mapping

to catalogue a list of metrics that can be used in software maintainability evaluation studies, specifically AOSM. Then, when researchers start their empirical studies, the time wasted looking for appropriated metrics to be used in their experiments will decrease. This paper is organized as follows. Section II describes the protocol used in this mapping study and in Section III we discuss the results obtained. An overview of the primary studies is also shown. The threats of validity are presented in Section IV and Section V presents conclusions and future work.

II. Systematic Mapping

Mapping Studies [13] try to gather all research related to a specific topic. Questions are broader and more general when compared to the ones present on Systematic Literature Reviews (SLRs) [14], for example: *What do we know about a topic T?*. This paper presents a systematic mapping study performed to identify which metrics are used to measure Aspect-Oriented (AO) and Object-Oriented (OO) software development regarding maintainability traits. We followed a strict protocol defined based on the guidelines proposed by Kitchenham and Charters [15]. Due to space constraints, we only show the most important parts of the protocol in the following sections. The complete protocol is available elsewhere [1] .

A. Research Questions

This study aims to query the literature and map primary studies that describe maintainability-specific metrics and their use to measure object-oriented (OO) and aspect-oriented (AO) software development. Thus, three research questions were defined, trying to find a representative metrics suite for our purpose:

1) **RQ1:** What metrics were adopted to assess software maintainability in AOSD?
2) **RQ2:** What metrics were adopted to assess software maintainability in OOSD?
3) **RQ3:** Which metrics that addressed OOSD maintainability can be adapted to address AOSD maintainability?

B. Data Source

The search strategy encompasses well-known digital library search engines. They were chosen based on the relevance for the computer science community and availability of papers for download. The Scopus digital library, for instance, was not included in this study because sometimes the papers were not available for downloading. The search process for this study was based on automated search using the following digital libraries:

- IEEE Computer Society Digital Library: http://ieeexplore.ieee.org;
- ACM Digital Library: http://dl.acm.org;
- EI Compendex: http://www.engineeringvillage2.org;
- Science Direct: http://www.sciencedirect.com/science.

[1] http://bit.ly/SM_AOSM

Since the search was performed on different days in different search engines, we decided to limit the final paper publication date to the 30th of June of 2011. The start date depended on where the paper came from. Thus, the initial publication year was 1992 for IEEE, 1993 for ACM and Science Direct, and 1996 for EI Compendex. Papers published after this date were not considered so as to produce a more homogeneous result and also to allow a future precise update of this study, which will consider publications after this date. The results of the search process are presented and discussed in Section III.

C. Primary Studies Search Strategy

This section describes the search strategy to select the primary studies. The first step was to build a search string. We used the PICOC (Population, Intervention, Comparison, Outcome, Context) [14] criteria to frame the search string. These criteria were derived from guidelines from the medical domain. Kitchenham and Charters [15] have adapted them to the context of software engineering. This approach helps researchers defining relevant terms based on key terms used by previous studies and expert insights. The resulting search string from this process was:

"Software Engineering" **AND** *("Aspect-Oriented Programming" OR "Maintainability" OR "Aspect-Oriented Software Development" OR "Crosscutting Concern" OR "Maintenance" OR "Object-Oriented Programming" OR "Object-Oriented Development" OR "Evolution")* **AND** *("Metrics" OR "Measurement" OR "Measure")*

After formulating the search string, a team of six researchers was composed to evaluate the search results: (i) 3 PhD students [Juliana Saraiva, Emanoel Barreiros, and Gustavo Pinto] and, (ii) 3 MSc students [Adauto Almeida, Aline Alencar, and Flavio Lima]. In addition, two professors supervised all the process, Sérgio Soares and Fernando Castor.

The selection of primary studies was conducted in three phases that are described below:

1) Selection of potentially relevant primary studies;
2) Evaluation of the results from the first selection against the inclusion/exclusion criteria;
3) Conflicts discussion and final selection.

Initially, only the title, keywords and abstracts were taken into account for paper inclusion. It is important to stress that only papers that were clearly out of scope were excluded in this phase. Then, all potential primary studies were kept for further analysis. As this activity is fairly simple, just **Juliana Saraiva** and **Emanoel Barreiros** performed it. The second phase considered the inclusion/exclusion criteria. Following the guidelines, some criteria should be proposed to select a more relevant set of papers, trying to keep only potentially relevant studies after the end of this phase. The inclusion/exclusion criteria were discussed with all researchers involved in this mapping. It is important to stress that the inclusion/exclusion criteria were validated by experts (Fernando Castor and Sergio Soares), based on their experience on the

studied topic. Next, the aforementioned exclusion criteria are listed:

- The paper is not real paper (presentation slides or extended abstracts);
- The paper is not related to software engineering;
- The paper does not present a maintainability/evolution metric;
- The paper does not present metric(s) related to OO or AO programming;
- The paper does not present the metric description.

On the second phase, three pairs of researchers were composed and all the potentially relevant papers were evenly distributed among them. Each paper was reviewed by at least two researchers: (i) *P01- Juliana and Gustavo*, (ii) *P02 - Adauto and Aline*, and (iii) *P03 - Emanoel and Flavio*. The result of this activity was the selection of the primary studies and it considered the exclusion criteria previously presented. Each researcher read the whole paper and made a list of studies that did not match any exclusion criteria. If a paper matched at least one of the 5 exclusion criteria, it was excluded from the mapping. During this part of the process, each pair built a report of agreements and disagreements regarding the permanence of each paper in the mapping study according to the evaluation criteria previously presented.

The third and last phase was the conflict resolution and final selection. A conflict resolution meeting was organized and the disagreements discussed under supervision of Sérgio Soares and Fernando Castor. In this final phase each researcher screened the full paper. The result of this meeting was the final set of primary studies. It is important to highlight that at the end of the process, a paper was selected if it had at least one maintainability metric, its description, and it should be related to OO or AO software. All primary studies received a unique identifier so that they could be easily referenced throughout the process. For instance, SM01 means: Systematic Mapping Study Number 01 (first paper selected). For each primary study, we recorded the following information:

- Reviewers (pair of researchers);
- Date of data extraction;
- Author(s);
- Journal/Conference where it was published;
- Year of publication;
- Metrics and their acronyms;
- Descriptions of the metrics and information on how to collect them;
- Paradigm (OO or AO).

III. Results Discussion

This section discusses the results found in this systematic mapping study. It is important to emphasize that the metrics presented in this section apply to just OOP and AOP.

A. Answers to Research Questions

This section presents the answers to the Research Questions raised by this systematic mapping.

TABLE I
Aspect-Oriented Maintainability Metrics

Metric	Description	Primary Studies that cited the metric
LCC	Loose Class Cohesion	SM3, SM37, SM68, SM76, SM181, SM209, SM219, SM236, SM330, SM331
CDO	Concern Diffusion over Operations	SM144, SM169, SM178, SM209, SM236, SM126, SM140
CAE	Coupling on Advice Execution	SM47, SM49, SM53, SM140, SM161, SM187, SM209
CDLOC	Concern Diffusion over Lines of Code	SM140, SM144, SM169, SM178, SM209, SM236
CDA	Crosscutting Degree of Aspects	SM49, SM53, SM140, SM161, SM209, SM236
CIM	Coupling on Intercepted Modules	SM53, SM140, SM161, SM177, SM209
CFA	Coupling on Field Access	SM49, SM53, SM161, SM177, SM187
WOM	Weighted Operations in Module	SM47, SM49, SM177, SM187
RFM	Response For a Module	SM49, SM53, SM161, SM329
CBM	Coupling Between Modules	SM47, SM49, SM92, SM177
VS	Vocabulary Size	SM144, SM209, SM350
UACOH	Unified Aspect Cohesion Metric	SM330, SM331
LCOO	Lack of Cohesion in Operations	SM330, SM331
DOS	Degree of scattering	SM126, SM169
CONC	Concentration	SM126, SM169

(1) RQ01: What metrics were adopted to assess software maintainability in AOSD? The goal of this question is to list metrics that can be used as indicators in the empirical research that assesses AOSM. Even though this topic is widely studied by many researchers, no catalogue encompassing most of the existing AO maintainability metrics was found. Thus, when researchers start their studies, they waste a certain amount of time looking for the appropriate metrics to be used or propose a metric that already exists, as discussed in Section III-B. Table I summarizes the answer to this question. In addition, Appendix A depicts each primary study by its identifier. A total of 67 AO maintainability metrics were found. However, due to space constraints, we decided to present the most relevant metrics. The relevance criterion was based on how many times each metric was cited by the primary studies. In this paper we only present 15 metrics that appeared at least twice in the selected papers. It means that each metric was used by at least one study that is not in the paper where the metric was presented. Nevertheless, it is important to notice that the complete catalogue comprising the 67 metrics is available at our research website [1] .

(2) RQ02: What metrics were adopted to assess software maintainability in OOSD? Despite the focus of this research in AOSM, since AOP might be considered an evolution of OOP, we took into account OO maintainability metrics to find which of them can be used as indicators in quantitative assessment of AOSM. From the 575 OO metrics listed, 37 (6.43%) already adhere to OOSD and AOSD, 69 (12%) cannot

be adapted to address AO specific programming structures and, 469 (81.58%) of them can be evolved. Nevertheless, just 19 metrics are depicted in Table II. Once more, due to space constraints, we were not able not show all the metrics we found. Based on the same relevance criterion we selected a number of metrics that we were able to present in the available number of pages. The metrics depicted here were referenced by eight or more primary studies. If our criterion was metrics referenced by seven or more primary studies, the number of metrics would be 23 and the page limit would not be respected. The complete set of metrics is available at the research website [1] .

(3) RQ03: Which metrics that addressed OOSD maintainability can be adapted to address AOSD maintainability? Actually, we expected that there were more metrics addressing OOSD than AOSD, since OOP is older and enjoys more widespread adoption than AOP. Consequently, this research question was raised trying to identify which of the existing OO metrics could be used, adapted and/or evolved to help in the AOSM analysis.

All OO metrics' descriptions were analyzed to check if it was possible to evolve them to adhere to AOP. We tried to identify OOSD language structures that were analogous to the ones present in AOSD languages, in order to define which OO metrics could be evolved to measure AOSD. Consequently, we considered that a class in OO languages is analogous to an aspect in AO languages. For instance, if there is a metric that counts the Number of Classes (NC) in an OO system, its adaptation to AO would count the number of aspects in an AO system (NAs). Methods in OO languages were considered analogous to advice in AO languages. For example, in OOP there is a metric that calculates the number of Weighted Methods of a Class (WMC) by counting the number of methods declared in a class. Again, its AO adaptation would represent the weight of an aspect and would compute the number of both methods and advice declared in an aspect. It was observed that most OO metrics could be adapted to AOP. Among the 575 OO metrics listed, 37 (6.43%) already adhere to both OOSD and AOSD, since LOC (Lines of Code), 69 (12%) cannot be adapted to address AO specific programming structures, such as, V (Volume) that is defined as the minimum possible volume for a given algorithm, and, 474 (82.43%) of them can be evolved. As example of a metric that can be evolved is TCC (Tight Cohesion Class) that can be adapted and turn into TCM (Tight Cohesion Modules), where modules can be classes and aspects.

Due to space constraints, Table III shows just 14 metrics. They should be cited by at least 6 primary studies and they must not have been adapted to address AOSD by previous work. There are some cases, such as NM (Number of Methods) that were cited by 8 primary studies, however there is a similar AO metric evolved from NM: NAd (Number of Advice). Because of this exclusion criterion to depict the metrics in this paper, the NM metric does not appear in Table III. The same cases occurred with RFC (Response For a Class) that was evolved to RFM (Response For a Module); WMC (Weight

TABLE II
OBJECT-ORIENTED MAINTAINABILITY METRICS

Metric	Description	Primary Studies that cited the metric
CBO	Coupling between Objects Classes	SM4, SM8, SM13, SM29, SM38, SM42, SM47, SM48, SM54, SM70, SM73, SM75, SM76, SM87, SM92, SM111, SM119, SM129, SM132, SM146, SM149, SM151, SM164, SM166, SM189, SM194, SM200, SM205, SM209, SM210, SM211, SM217, SM222, SM266, SM301, SM313, SM319, SM329, SM341, SM343, SM346, SM347, SM156
NOC	Number of Children of a Class	SM4, SM7, SM13, SM31, SM35, SM39, SM42, SM48, SM49, SM53, SM54, SM66, SM69, SM70, SM75, SM76, SM79, SM87, SM132, SM156, SM164, SM187, SM189, SM194, SM200, SM201, SM204, SM205, SM209, SM210, SM217, SM222, SM224, SM266, SM290, SM313, SM319, SM325, SM347
LCOM	Lack of Cohesion in Methods	SM1, SM3, SM7, SM13, SM35, SM37, SM39, SM48, SM54, SM69, SM70, SM75, SM76, SM87, SM105, SM111, SM123, SM128, SM146, SM164, SM165, SM189, SM194, SM200, SM210, SM217, SM222, SM226, SM66, SM290, SM307, SM313, SM319, SM325, SM330, SM331, SM335, SM345, SM347
DIT	Depth of Inheritance Tree	SM123, SM4, SM7, SM13, SM31, SM48, SM49, SM54, SM69, SM70, SM75, SM76, SM87, SM96, SM132, SM144, SM164, SM177, SM187, SM189, SM194, SM200, SM201, SM204, SM205, SM209, SM210, SM217, SM222, SM22, SM266, SM290, SM313, SM319, SM325, SM347, SM39, SM42, SM66
RFC	Response For a Class	SM4, SM13, SM38, SM39, SM47, SM48, SM54, SM69, SM70, SM73, SM75, SM76, SM87, SM111, SM123, SM129, SM132, SM146, SM156, SM164, SM189, SM194, SM200, SM205, SM209, SM210, SM217, SM222, SM266, SM313, SM319, SM325, , SM329, SM343, SM346, SM347
WMC	Weighted Methods Per Class	SM7, SM8, SM31, SM35, SM39, SM48, SM54, SM69, SM70, SM75, SM87, SM111, SM123, SM132, SM138, SM146, SM156, SM164, SM189, SM194, SM200, SM201, SM204, SM210, SM217, SM222, SM266, SM290, SM313, SM319, SM325, SM337, SM347
LOC	Lines of Code	SM7, SM10, SM23, SM47, SM49, SM74, SM111, SM138, SM140, SM144, SM146, SM156, SM176, SM195, SM203, SM208, SM109, SM217, SM219, SM236, SM269, SM275, SM293, SM323, SM327
MPC	Message Passing Coupling	SM13, SM38, SM54, SM66, SM73, SM76, SM132, SM146, SM151, SM166, SM209, SM301, SM319, SM341, SM343, SM346, SM234
CC	Class Coupling	SM7, SM23, SM47, SM54, SM69, SM86, SM138, SM181, SM198, SM199, SM203, SM204, SM208, SM269, SM288, SM335
DAC	Data Abstraction Coupling	SM4, SM13, SM31, SM35, SM54, SM76, SM129, SM146, SM151, SM209, SM319, SM341, SM343, SM346
TCC	Tight Class Cohesion	SM3, SM31, SM37, SM68, SM76, SM105, SM111, SM146, SM176, SM181, SM219, SM277, SM335
LCC	Loose Class Cohesion	SM3, SM37, SM68, SM76, SM181, SM209, SM219, SM236, SM330, SM331
LCOM3	Lack of Cohesion in Methods 3	SM3, SM37, SM68, SM176, SM181, ,SM219, SM267, SM277
NC	Number of Classes	SM96, SM203, SM82, SM226, SM230, SM288, SM136
LCOM2	Lack of Cohesion in Methods 2	SM3, SM37, SM68, SM176, SM182, SM219, SM224, SM267, SM277
LCOM1	Lack of Cohesion in Methods 1	SM3, SM7, SM68, SM176, SM181, SM219, SM224, SM264, SM267, SM277
NOM	Number of Methods	SM15, SM35, SM54, SM76, SM111, SM146, SM290, SM319
NM	Number of Methods	SM70, SM82, SM96, SM136, SM204, SM224, SM230, SM288
NA	Number of Attributes	SM10, SM96, SM136, SM144, SM203, SM204, SM205, SM208
CDC	Concern Diffusion over Components	SM126, SM140, SM144, SM169, SM178, SM209, SM236, SM325

TABLE III
OBJECT-ORIENTED MAINTAINABILITY METRICS THAT CAN BE EVOLVED/ADAPTED

Metric	Description	Primary Studied that cited the metric
CBO	Coupling between Objects Classes	SM4, SM8, SM13, SM29, SM38, SM42, SM47, SM48, SM54, SM70, SM73, SM75, SM76, SM87, SM92, SM111, SM119, SM129, SM132, SM146, SM149, SM151, SM164, SM166, SM189, SM194, SM200, SM205, SM209, SM210, SM211, SM217, SM222, SM266, SM301, SM313, SM319, SM329, SM341, SM343, SM346, SM347, SM156
NOC	Number of Children of a Class	SM4, SM7, SM13, SM31, SM35, SM39, SM42, SM48, SM49, SM53, SM54, SM66, SM69, SM70, SM75, SM76, SM79, SM87, SM132, SM156, SM164, SM187, SM189, SM194, SM200, SM201, SM204, SM205, SM209, SM210, SM217, SM222, SM224, SM266, SM290, SM313, SM319, SM325, SM347
DAC	Data Abstraction Coupling	SM4, SM13, SM31, SM35, SM54, SM76, SM129, SM146, SM151, SM209, SM319, SM341, SM343, SM346
TCC	Tight Class Cohesion	SM3, SM31, SM37, SM68, SM76, SM105, SM111, SM146, SM176, SM181, SM219, SM277, SM335
LCOM1	Lack of Cohesion in Methods 1	SM3, SM7, SM68, SM176, SM181, SM219, SM224, SM264, SM267, SM277
LCOM2	Lack of Cohesion in Methods 2	SM3, SM37, SM68, SM176, SM182, SM219, SM224, SM267, SM277
LCOM3	Lack of Cohesion in Methods 3	SM3, SM37, SM68, SM176, SM181, ,SM219, SM267, SM277
LCOM5	Lack of Cohesion in Methods 5	SM3, SM37, SM176, SM181, SM219, SM267, SM277
LCOM4	Lack of Cohesion in Methods 4	SM3, SM37, SM176, SM181, SM2667, SM277
COF	Coupling Factor	SM11, SM38, SM198, SM211, SM343, SM346

Methods Per Class) that was adapted and named as WOM (Weight Operations Per Module); CBO (Coupling Between Objects) that was turned into CBM (Coupling Between Objects) and, others that can be saw at our website [1] .

In addition, an in-depth study should be carried out to investigate which AO-specific syntactic structures could be assessed to propose new maintainability metrics to adhere to AOP. As this work is an exploratory study, for all the metrics that we considered adaptable it is necessary to conduct an investigation to check and validate the efficiency of the adaptation of OO metrics to AO. It is possible to observe, analyzing Tables I and II that some metrics have already been used in both OOSM and AOSM assessment studies, since thay are in both tables.

B. Discussion on Identified Metrics

The maintainability metrics identified by this mapping study are faced as candidates to compose a metrics suite for helping researchers in empirical studies. With a metrics suite in mind, researchers can more easily choose the ones that better fit their intent when designing empirical studies (either quantitative or qualitative). The number of times each metric was cited is a relevance measure that can also be used by researchers when looking into the whole set of 610 metrics. They can use a subset of these metrics in their empirical evaluations as indicators of the software quality level regarding software maintenance. A total of 610 metrics were found, 575 related to OO and 67 related to AO. It is important to clarify that 32 metrics adhere both to OO and AO. Considering this total, 94.4% are related to OOSM. This result was already expected since AO is less mature than OO. Also, some researchers consider that AO is an evolution over OO [8], which may lead to the conclusion that many of the metrics associated with OO may also be used in the AO context.

In this paper only 35 metrics were shown, 20 OO metrics and 15 AO metrics. This occurred because we had to choose relevance criteria to select which of them would be depicted in this paper, considering space constraints. The relevance criteria were (i) for AO metrics, two or more primary studies cited the metric and, (ii) for OO metrics, eight or more primary studies cited the metric. For OO metrics, we could show only metrics that had eight or more citations, if we show metrics that have seven or more citations, we would not have space to depict them (23 metrics). The number of citations is higher for OO metrics because we found a higher number of metrics for this paradigm. Thus, we had to be more rigorous for this group of metrics. Nevertheless, all the metrics can be found at the website [1] previously mentioned.

The first eight metrics depicted in Tables I and II were mentioned by a large number of papers. However, it is important to clarify that all those metrics found in this study were catalogued and considered as an achievement. Thus, researchers have successfully used this group of metrics in several empirical evaluations. As a consequence, they can be considered as part of a metric suite related to software maintainability analysis. However, it is important to clarify that all the other metrics were not disregarded by this study, they all were catalogued. Another point to raise is that most of these OO metrics can be used in the AOSM investigation because nearly all syntactic structures found in OOP languages have analogous structures in AOP languages.

Metrics with different names and the same meanings is another important issue to point out. For instance, DIT (Depth of Inheritance Tree) and DIH (Depth of Inheritance) are metrics that are defined as the maximum length from the node to the root of the tree. This means that they measure the number of ancestors of a class/aspect. This is similar to NOA (Number of Ancestors) and NAC (Number of Ancestors) as well to TLOC (Total Lines of Code) and LOC (Lines of Code). On the other hand, there are metrics with the same name and different meanings, such as NA, which can be Number of Ancestor or Number of Attributes. These situations occurred because there is no convention for naming software metrics.

It is important to highlight the topics related to maintainability metrics that were addressed by all primary stud-

TABLE IV
TOPICS COVERED BY THE METRICS FOUND

Topic	# Citations	Total Percentage of Metrics
Size	150	24.6%
Cohesion	223	36.5%
Coupling	170	27.8%
Inheritance	60	9.84%
Software Architecture Constraint	7	1.26%

TABLE V
EVOLUTION OF NUMBER OF PAPERS INCLUDED

Source	#Papers	1st. Selec.	2nd. Selec.	% Included	Interval
IEEE	2180	225	83	3.8%	1982 - 2011
ACM	2386	119	41	1.7%	1989 - 2011
EI Compendex	173	50	25	14.4%	1969 - 2011
Science Direct	881	26	16	1.8%	1985 - 2011
TOTAL	5620	423	165	2.93%	1969 - 2011
TOTAL*	5175	351	138	2.66%	1969 - 2011

*Number of papers after the removal of all duplicates.

ies: Software Architecture Constraints, Inheritance, Cohesion, Coupling, and Size. Table IV shows the results. The first column presents the topics related to maintainability described by the primary studies. The number of maintainability metrics that claimed to be investigating the topic and relating it to the topics is shown in the second column. Finally, the third column shows what is the percentage of the number of metrics in the second column considering the total of 610 maintainability metrics found. We discovered these topics by reading and performing textual search on the descriptions of the metrics and afterwards discussing the obtained results. Thus, if a metric description contains information about cohesion, we considered that this metric is directly related to the cohesion topic. This occurred for inheritance, coupling, software architecture constraints, and size. For instance, there is a metric called AIM (All Inherited Methods) that means the average of local and overridden/inherited methods in a system. So, the topic associated with this metrics was **Inheritance**.

Table IV shows that size, coupling and cohesion are the most commonly investigated topics addressing software maintainability in the literature. More than 89% of the papers mentioned these topics, providing a clear relationship between them and maintainability metrics. And finally, Inheritance and software architecture constraint appears as another important topics directly related to software maintainability among the primary studies.

C. Interesting Findings

In addition to the main contribution of the systematic mapping study presented here (the metrics suite that can be used to assess AOSM) this section presents some results obtained during the execution of the study.

At the end of the selection process, 138 primary studies were selected. However, just 117 papers were presented here. These papers are the primary studies that appeared in Tables I, II and III. Appendix A shows the selected primary studies. The complete list of primary studies involved can be found at our research website [1] .

1) Digital Libraries: During the systematic mapping some papers have been excluded according to the previously described mapping protocol. Table V shows the results of each phase. The first column presents the digital libraries. The second column represents the total number of papers initially returned. The third column depicts the result of the first selection, which consisted of evaluating the paper's title, abstract and keywords. The fourth column exposes the number of papers included after the second phase of analysis, which consisted of screening the full paper. The fifth column indicates the percentage of papers included, considering the initial number of papers returned, and the final selection. The last column shows the interval of primary studies publication after the initial query. The last row presents the number of papers after removing all duplicates. For instance, if a paper was found on both ACM and IEEE digital libraries, it was accounted and considered just once. The other occurrence would be marked as a duplicate and removed from the mapping.

Analyzing Table V, it is possible to see that EI Compendex had the best performance, with 14.4% of papers included. Despite having the lowest number of papers, EI Compendex could provide a more accurate list of papers when compared to the other search engines, adding less noise. The ACM digital library had the worst performance, demanding more work in the selection of the papers. Another point we observed is the low number of selected papers from it.

Table V also shows that, among the 5175 papers returned by the digital libraries, only 138 were selected. We observed that the queries presented a considerable level of noise, since only 2.7% of the papers were actually relevant to the mapping. Many factors can contribute to increase the noise, e.g., we might have not used the ideal set of keywords. Kitchenham and Charters have already discussed the problem in using automated search engines like the ones we employed in this mapping [15]. One more time, EI Compendex appears prominently. It contains the oldest paper returned in this study. It is important to notice that after the paper exclusion process, the oldest paper selected was from 1992 and the most recent one was published on June, 2011. This range shows that maintainability metrics have been studied and proposed for at least 19 years. Even though software maintainability metrics have been a research topic through all these 19 years, there is no catalogue for them.

2) Authors Cited: The authors who wrote the primary studies are presented in this section. Table VI shows the number of authors who wrote about software maintainability metrics. The first column indicates how many times a given researcher appears as author of a primary study.

As depicted by Table VI, Lionel C. Briand is the researcher who published more papers on the topic on which this mapping study is interested. It is important to highlight that most of the papers found are only using metrics to perform quantitative

TABLE VI
AUTHORS WHO WROTE ABOUT AOSM METRICS

#Papers	Authors
1	All the others.
2	Alfred Aho, Andrea De Lucia, Bandar Alshammari, Baowen Xu, Colin Fidge, Diane Corney, Doo Hwan Bae, Erik Arisholm, Esperanza Manso, F. Fioravanti, Fernando Castor, Gail C. Murphy, Hany H. Ammar, Jianjun Zhao, Jonas Lundberg, Letha Etzkom, Marc Eaddy, Mikael Lindvall, Nelio Cacho, R. Harrison, Santonu Sarkar, Thais Batista, Victor R. Basili, Welf Lowe, Yuming Zhou.
3	Claudio Sant'Anna, Denys Poshyvanyk, Doo Hwan Bae, Ewan Tempero, Heung Seok, Chae, Jehad Al Dallal, John W. Daly, Jurgen K. Wust, P. Nesi, R. Nithi, S. Counsell.
4	Avadhesh Kumar, Marcela Genero, Mario Piattini, P.S. Grover, Rajesh Kumar.
5	Alessandro Garcia.
8	Lionel C. Briand.

assessments. This means that the minority of the papers we selected in this mapping actually propose new metrics.

3) Journal and Conferences: Here we show the journals and conferences that published the papers about software maintainability metrics that were selected by the mapping. The Journals are depicted below, organized by the number of occurrences of primary studies. The TSE - IEEE Transactions on Software Engineering journal is the vehicle that has published more papers concerning software maintainability metrics. This means that, after all exclusion processes and analysis of the remaining papers, this journal provided the highest number of relevant papers.

- **27 Occurrences:** TSE;
- **10 Occurrences:** INFSOF;
- **4 Occurrences:** JSS;
- **3 Occurrences:** Information Sciences;
- **2 Occurrences:** Sciences of Computer Programming, SMR;
- **1 Occurrence:** ESE, IJSEKE, TOSEM, JSA, SQJ.

Regarding conferences, METRICS had the highest number of published papers. This result is expected since this conference is completely related to the subject assessed in this systematic mapping. Because of space limitations, only the conferences' acronyms are presented. Next are the results of the number of papers published on the proceedings of each conference:

- **8 Occurrences:** METRICS;
- **7 Occurrences:** CSMR, ICSM;
- **4 Occurrences:** APSEC, ASWEC;
- **3 Occurrences:** ACM - SIGSOFT Conference;
- **2 Occurrences:** AOSD, ESEM, STEP, WETSoM;
- **1 Occurrence:** ACE, ACoM, ACSC, ASE, ASEW, CASCON, CSSE, CW, CYCSYN, ECSA, ESM, FSE, HIS, ICC, ICCRD, ICECCS, ICETEC, ICIW, ICPC, INFOS, INMIC, ISESE, ISSTA, MOMPES, PerCom, KAMW, QSIC, REV, SAC, SBES, SEAA, SEM SEW, SIGPLAN, SNPD, SOQUA, TASE, TOOLS, WCIT, WoSQ, WSCS, ACM-SE, ACM-CSUR, ACM-SIGAda.

CSMR and ICSM come close to METRICS in number of papers published on software maintainability metrics, which is expected, since these two conferences are directly related to software maintainability. Another important insight is that most papers were published in conferences. Among the 138 selected papers, 63 (45.7%) were published in journals and 75 (54.3%) papers were published in conference proceedings. We believe that this result could be by the larger number of conferences and workshops, when compared to the number of Software Engineering journals. Moreover, it is well-known that the time between the initial submission and the publication of a paper in a journal is longer. However, an in-depth study should be conducted.

IV. THREATS TO VALIDITY

The first threat to validity is related to the search strategy employed. Since we mainly used automated search engines, relevant studies may not have been included in the set of selected studies. Even though we dedicated some time to identify relevant keywords, a particular study that used a different term, but yet relevant, might be missing.

The second threat observed is that some more recent studies may be missing because the search engines may not have indexed them. Other systematic mapping study should be carried out, with the same research string, trying to find primary studies that were not contemplated here. The small number of selected papers, when compared with the number of returned papers from the digital libraries, is also a threat. Even though the automated search has been executed without any time limitation, only 138 papers were considered relevant. Specifically for this systematic mapping, the existence of the metric description in the primary study as an inclusion criterion might have caused the removal of some potentially relevant papers.

V. CONCLUDING REMARKS

This paper presents a systematic mapping on aspect-oriented software maintainability metrics. The extraction process used in this systematic mapping was detailed. Based on that, other researchers can better evaluate it and eventually try to reproduce it. It is important to notice that the participation of eight people in this mapping (three MSc students, three PhD students and two professors) is an important mechanism to try to reduce the evaluation bias. This diversity of opinions caused conflicts and discussions about the ideal metrics suite to be shown as the final result of this mapping study. We also show all the information about the digital libraries, authors, conferences and journals that addressed the subject analyzed in this study. EI Compendex was the most effective digital library used in this systematic mapping study. METRICS Conference is the conference that published more papers related to software maintainability metrics. In addition, TSE - IEEE Transactions on Software Engineering Journal published a higher number of primary studies selected, 27 occurrences. Interestingly, most of the selected papers just proposed the

metrics. This result was concluded because from the 610 metrics found, just 91 (15%) had two or more citations in different primary studies. Thus, we inferred that very few studies actually used the metrics suites previously proposed.

Metrics can be used as indicators in the software assessments, supporting both quantitative and qualitative studies. In our previous research, we were not able to find any other work that lists such an extensive number of metrics like the current mapping. Analyzing the results, it is possible to notice that the majority of metrics addressing software maintainability were related to software cohesion, coupling and inheritance. Also, there were more metrics addressing object-oriented software maintainability (OOSM) than aspect-oriented software maintainability (AOSM), as expected. This occurred because OO has been studied for more time than AO. OO is also far more adopted in industry when compared to AO. Consequently, there are more studies and metrics related to this "paradigm". In addition, it is possible to mention that a great number of OO metrics can be adapted or evolved to address AO.

As future work, as we found a large number of metrics, we plan check the tools available to collect the OO and AO metrics automatically, both for object-oriented and aspect-oriented software development. Case studies using the metrics suite shown in this work to validate their potential when adopted in aspect-oriented software quantitative assessments are another task to be done in the forthcoming steps of our research. A backward search on the primary studies' references can be performed to find other possible metrics that were not included in this work because of, for example, some missing relevant keywords on the search string. There were some cases where the paper was excluded because it presented only the metric acronym, however its description was in a referenced paper. In-depth studies can be done to find more maintenance metrics. The analysis of other concerns that could have relation with software maintainability is needed. Only software size, cohesion, coupling, inheritance and software architecture constraints were identified. Another point that still requires investigations is the adaptation of OO metrics to adhere to AO. A deep investigation to check and validate the efficiency of the adaptation of OO metrics to AO is demanded. So, an exploratory study to assess the new proposal of AO metrics suite derived from OO metrics should be done.

ACKNOWLEDGMENTS

Juliana and Emanoel are supported by FACEPE. Sergio is partially supported by CNPq and FACEPE, grants 305085/2010-7 and APQ-0093-1.03/08. Fernando is supported by CNPq (306619/2011-3 and 475157/2010-9), FACEPE (APQ-0395-1.03/10), and by INES (CNPq 573964/2008-4 and FACEPE APQ-1037-1.03/08). This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08. We would like to thank the anonymous referees, who helped to improve this paper with insightful comments and suggestions.

REFERENCES

[1] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering: an introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[2] W. F. Tichy, *Should Computer Scientists Experiment More?* Los Alamitos, CA, USA: IEEE Computer Society, 1997, vol. 31.

[3] V. R. Basili, "The role of experimentation in software engineering: past, current, and future," in *ICSE'96: Proceedings of the 18th international conference on Software engineering*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 442–449.

[4] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in *ESEM'09: Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009.

[5] F. Silva, A. Santos, S. S., C. Franca, C. Monteiro, and F. Maciel, "Six years of systematic literature reviews in software engineering: An updated tertiary study," *International Journal Information and Software Technology*, pp. 889–913, 2011.

[6] A. Almeida, E. Barreiros, J. Saraiva, and S. Soares, "Mecanismos para guiar estudos empiricos em engenharia de software: Um mapeamento sistematico. (in portuguese)," in *ESELAW'11: Proceedings of the 8th Experimental Software Engineering Latin America Workshop*, 2011.

[7] E. Barreiros, A. Almeida, J. Saraiva, and S. Soares, "A systematic mapping study on software engineering testbeds," in *ESEM'11: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement*, 2011.

[8] M. Eaddy, A. Aho, and G. C. Murphy, "Identifying, assigning, and quantifying crosscutting concerns," in *ACoM'07: Proceedings of the 1st ACoM Workshop*, 2007.

[9] C. SantAnna, A. Garcia, C. Chavez, C. Lucena, and A. Staa, "On the reuse and maintenance of aspect-oriented software: An assessment framework," in *SBES'10: Proceedings of the Brazilian Symposium of Software Engineering*, 2003.

[10] J. Saraiva, S. Soares, and F. Castor, "Assessing the impact of aosd on layered software architectures," in *ECSA'10: Proceedings of the 4th European Conference on Software Architecture*, 2010.

[11] ——, "A metrics suite to evaluate the impact of aosd on layered software architectures," in *ESCOT'11: Proceedings of the Empirical Evaluation of Software Composition and Techniques*, 2011.

[12] ——, "Analyzing architectural conformance of layered aspect-oriented systems with arche meter," in *AOSD'11: Proceedings of the 10th Annual International Conference on Aspect-Oriented Software Development*, 2011.

[13] H. Arksey and L. O'Malley, "Scoping studies: towards a methodological framework," *International Journal of Social Research Methodology*, vol. 8, no. 1, pp. 19–32, 2005.

[14] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Tech. Rep., 2004.

[15] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Software Engineering Group, School of Computer Science and Mathematics, Keele University, Tech. Rep., July 2007.

APPENDIX A – THE PRIMARY STUDIES

[SM1] Heung Seok Chae, Yong Rae Know. A cohesion measure for classes in objectoriented systems. METRICS'98.

[SM3] Letha H. Etzkorna, Sampson E. Gholstonb, Julie L. Fortuneb, Cara E. Steina, Dawn Utleyb, Phillip A. Farringtonb, Glenn W. Coxa. A comparison of cohesion metrics for objectoriented systems. INFSOF'04.

[SM4] Aaron B. Binkley, Stephen R. Schach. A comparison of sixteen quality metrics for objectoriented design. IS'96

[SM7] Hashem Yazbek. A concept of quality assurance for metrics in CASE tools. SIGSOFT'10.

[SM8] Camelia Serban. A Conceptual Framework for Objectoriented Design Assessment. ESM'10.

[SM10] Vu Nguyen, Barry Boehm, Phongphan Danphitsanuphan. A controlled experiment in assessing and estimating software maintenance tasks. INFSOF'10.

[SM13] Youssef Hassoun, Roger Johnson, Steve Counsell.A dynamic runtime coupling metric for metalevel architectures. CSMR'04.

[SM15] Jagdish Bansiya, and Carl G. Davis. A hierarchical model for objectoriented design quality assessment. TSE'02.

[SM23] Tatsuya Miyake, Yoshiki Higo, Katsuro Inoue. A metricbased approach for reconstructing methods in objectoriented systems. WoSQ'08.

[SM29] Huan Li. A Novel Coupling Metric for ObjectOriented Software Systems. Transaction on IEEE'08.

[SM31] Thomas Panas, Rudiger Lincke, Jonas Lundberg, Welf Lowe. A Qualitative Evaluation of a Software Development and ReEngineering Project. SEW'05.

[SM35] Mehwish Riaz, Emilia Mendes, Ewan Tempero. A systematic review of software maintainability prediction and metrics. ESEM'09.

[SM37] Lionel C. Briand, John W. Daly, and Jurgen K. Wust. A unified framework for cohesion measurement in Objec Oriented systems. TSE'97.

[SM38] Lionel C. Briand, John W. Daly, and Jurgen K. Wust. A unified framework for coupling measurement in objectoriented systems. TSE'99.

[SM39] Victor R. Basili, Fellow, IEEE, Lionel C. Briand, and Walcelio L. Melo. A validation of objectoriented design metrics as quality indicators. TSE'96.

[SM42] Jill Doake, Ishbel Duncan. Amber metrics for the testing and maintenance of objectoriented designs. CSMR'98.

[SM47] Jennifer Munnelly, Serena Fritsch, Siobhan Clarke. An AspectOriented Approach to the Modularisation of Context. PerCom'07.

[SM48] Ayaz Farooq, Rene Braungarten, Reiner R. Dumke. An Empirical Analysis of ObjectOriented Metrics for Java Technologies. INMIC'05.

[SM49] Rachel Burrows, Fabiano Cutigi Ferrari, Alessandro Garcia, Francois Taiani. An Empirical Evaluation of Coupling Metrics on AspectOriented Programs. WETSoM'10.

[SM53] Haihao Shen, Sai Zhang, Jianjun Zhao. An Empirical Study of Maintainability in AspectOriented System Evolution Using Coupling Metrics. TASE'08.

[SM54] W. Lia, L. Etzkorna, C. Davisa, J. Talburt. An empirical study of objectoriented system evolution. INFSOF'00.

[SM66] F.G. Wilkiea, B.A. Kitchenham. An investigation of coupling, reuse and maintenance in a commercial C++ application. INFSOF'01.

[SM68] Jehad Al Dallal, Lionel C. Briand. An objectoriented highlevel designbased class cohesion metric. INFSOF'10.

[SM69] William W. Pritchett IV. An objectoriented metrics suite for Ada 95. SIGAda'01.

[SM70] R. Harrison, S. Counsell, R. Nithi. An overview of objectoriented design metrics. STEP'97.

[SM73] M.K Abdi, H. Lounis, H. Sahraoui. Analyzing Change Impact in ObjectOriented Systems. SEAA'06.

[SM74] Santonu Sarkar, Girish Maskeri Rama, and Avinash C. Kak. APIBased and InformationTheoretic Metrics for Measuring the Quality of Software Modularization. TSE'07.

[SM75] Ghassan Allcadi, Doris L. Carver. Application of metrics to objectoriented designs. TSE'98.

[SM76] Usha Kumari, Sucheta Bhasin. Application of objectoriented metrics To C++ and Java a comparative study. SIGSOFT'11.

[SM79] Juha Gustafsson, Jukka Paakki, Lilli Nenonen, and A. Inkeri Verkamo. Architecturecentric software evolution by software metrics and design patterns. CSMR'05.

[SM82] Juliana Saraiva, Sergio Soares, Fernando Castor. Assessing the impact of AOSD on layered software architectures. ECSA'10.

[SM84] Bandar Alshammari, Colin Fidge, Diane Corney. Assessing the impact of refactoring on securitycritical object-oriented designs. APSEC'10.

[SM86] Ebrahim Bagheri, Dragan Gasevic. Assessing the maintainability of software product line feature models using structural metrics. SQJ'10.

[SM87] Giulio Concas, Michele Marchesi, Alessandro Murgia, Sandro Pinna, Roberto Tonelli.Assessing traditional and new metrics for objectoriented systems. WETSoM'10.

[SM92] Mikael Lindvall, Roseanne Tesoriero, Patricia Costa. Avoiding architectural degeneration an evaluation process for software architecture. METRICS'02.

[SM96] Marcela Genero, Mario Piattini, Esperanza Manso, Giovanni Cantone. Building UML class diagram maintainability prediction models based on early metrics. METRICS'03.

[SM105] Sami Makela, Ville Leppanen. Clientbased cohesion metrics for Java programs. SCP'09.

[SM109] Mikhail Perepletchikov, Caspar Ryan, and Keith Frampton. Cohesion Metrics for Predicting Maintainability of ServiceOriented Software. QSIC'07.

[SM111] Rudiger Lincke, Jonas Lundberg and Welf Lowe. Comparing software metrics tools. ISSTA'08.

[SM119] R. Harrison, S. Counsell, R. Nithi. Coupling metrics for objectoriented design. METRICS'98.

[SM123] M. Ajmal Chaumun, Hind Kabaili, Rudolf K. Keller, Francois Lustman and Guy SaintDenis. Design properties and objectoriented software changeability. CSMR'00.

[SM126] Marc Eaddy, Thomas Zimmermann, Kaitlin D. Sherwood, Vibhav Garg, Gail C. Murphy, Nachiappan Nagappan, Alfred V. Aho. Do Crosscutting Concerns Cause Defects? TSE'08.

[SM127] T.H. Ng, S.C. Cheung, W.K. Chan and Y.T. Yu. Do Maintainers Utilize Deployed Design Patterns Effectively? ICSE'07

[SM128] Varun Gupta, Jitender Kumar Chhabra. Dynamic cohesion measures for objectoriented software. JSA'11.

[SM129] Erik Arisholm, Lionel C. Briand, Audun Fyen. Dynamic coupling measurement for objectoriented software. TSE'04.

[SM132] Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson. Dynamic metrics for object-oriented designs. METRICS'99.

[SM136] Marcela Genero, Mario Piattini and Coral Calero. Empirical validation of class diagram metrics. ISESE'02.

[SM138] Fabrizio Fioravanti, Paolo Nesi. Estimation and prediction metrics for adaptive maintenance effort. TSE'01.

[SM140] Guadalupe Ortiz, Behzad Bordbar, Juan Hernandez. Evaluating the use of AOP and MDA in web service development. ICIW'08.

[SM144] Fernando Castor Filho, Nelio Cacho, Eduardo Figueiredo, Raquel Maranhao, Alessandro Garcia, Cecilia Mary F. Rubira.Exceptions and aspects the devil is in the details. FSE'06.

[SM146] Lalji Prasad,Aditi Nagar. Experimental Analysis of Different Metrics (ObjectOriented and Structural of Software). CYCSYN'09.

[SM149] V. Krishnapriya, K. Ramar. Exploring the Difference Between Object-Oriented Class Inheritance and Interfaces Using Coupling Measures. ACE'10.

[SM151] Lionel C. Briand, Jurgen Wu, John W. Daly, D. Victor Porter. Exploring the relationships between design measures and software quality in objectoriented systems. JSS'00.

[SM156] Horst Zuse. Foundations of objectoriented software measures. METRICS'96.

[SM161] Avadhesh Kumar, Rajesh Kumar, P.S. Grover. Generalized Coupling Measure for AspectOriented Systems. SIGSOFT'09.

[SM164] Bruno Stiglic, Marjan Heri6ko, Ivan Rozlnan. How to evaluate objectoriented software development? SIGPLAN'05.

[SM165] Yuming Zhou, Baowen Xu, Jianjun Zhao, Hongji Yang. ICBMC an improved cohesion measure for classes. ICSM'02.

[SM166] Gabriele Bavota, Andrea De Lucia, Rocco Oliveto. Identifying Extract Class refactoring opportunities using structural and semantic cohesion measures. JSS'10.

[SM169] Marc Eaddy, Alfred Aho, Gail C. Murphy. Identifying, Assigning, and Quantifying Crosscutting Concerns. ACoM'07.

[SM176] Heung Seok Chae, Yong Rae Kwon, Doo Hwan Bae. Improving cohesion metrics for classes by considering dependent instance variables. TSE'04.

[SM177] Mario Luca Bernardi, Giuseppe Antonio Di Lucca. Improving Design Pattern Quality Using Aspect Orientation. STEP'05.

[SM178] Nelio Cacho, Thais Batista, Alessandro Garcia, Claudio SantAnna, Gordon Blair. Improving Modularity of Reflective Middleware with AspectOriented Programming. SEM'06.

[SM181] Jehad Al Dallal. Improving the applicability of objectoriented class cohesion metrics. INFSOF'11.

[SM182] Charles H. House. Information Worker Tools Selection, Adoption and Evaluation Lessons from Software Development history. ICSS'05

[SM187] MATHUPAYAS THONGMAK, PORNSIRI MUENCHAISRI. Maintainability metrics for aspectoriented software . IJSEKE'09.

[SM189] Shyam R. Chidamber, David P. Darcy, Chris F. Kemerer. Managerial use of metrics for objectoriented software an exploratory analysis. TSE'98.

[SM193] Lionel C. Briand, Sandro Morasca, Victor R. Basili. Measuring and assessing maintainability at the end of high level design. TSE'93.

[SM194] AhRim Han, SangUk Jeon, DooHwan Bae, JangEui Hong. Measuring behavioral dependency for improving changeproneness prediction in UMLbased design models. JSS'10.

[SM195] P.S. Grover, Rajesh Kumar, Avadhesh Kumar. Measuring Changeability for Generic AspectOriented Systems. SIGSOFT'08.

[SM198] Tobias Mayer, Tracy Hall. Measuring OO systems a critical analysis of the MOOD metrics. TOOLS'99.

[SM199] Tianlin Zhou, Baowen Xu, Liang Shi, Yuming Zhou, Lin Chen. Measuring Package Cohesion Based on Context. WSCS'08.

[SM200] G. Manduchi, C. Taliercio. Measuring software evolution at a nuclear fusion experiment site A test case for the applicability of OO. INFSOF'02.

[SM201] Philippe LiThiaoTt, Jessie Kennedy and John Owens. Mechanisms for interpretation of OO systems design metrics. TSE'98.

[SM203] P. Nesi, M. Campanai. Metric framework for objectoriented realtime systems specification languages. JSS'96.

[SM204] G. Bucci, F. Fioravanti, P. Nesi, S. Perlini. Metrics and tool for system assessment. ICECCS'98.

[SM205] N. Debnath, L. Baigorria, D. Riesco,G. Montejano. Metrics applied to Aspect Oriented Design using UML profiles. Transaction on IEEE'08.

[SM208] F. Fioravanti, P. Nesi, F. Stortoni. Metrics for controlling effort during adaptive maintenance of object-oriented systems. ICSM'99.

[SM209] Tassia A. V. Freitas, Thais V. Batista, Flavia C. Delicato, Paulo F. Pires. Metrics for Evaluation of AspectOriented Middleware. SBES'09.

[SM210] Frederick T. Sheldon, Kshamta Jerath, Hong Chung. Metrics for maintainability of class inheritance hierarchies. SMR'02.

[SM211] Santonu Sarkar, Avinash C. Kak, Girish Maskeri Rama. Metrics for Measuring the Quality of Modularization of LargeScale ObjectOriented Software. TSE'08.

[SM217] Shen Zhang, Yongji Wang, Junchao Xiao. Mining Individual Performance Indicators in Collaborative Development Using Software Repositories. APSEC'08.

[SM219] Yixun Liu, Denys Poshyvanyk, Rudolf Ferenc, Tibor Gyimothy, Nikos Chrisochoides. Modeling class cohesion as mixtures of latent topics. ICSM'09.

[SM222] Dapeng Liu, Shaochun Xu. New Quality Metrics for ObjectOriented Programs. SNPD'07.

[SM224] Sahar R. Ragab, Hany H. Ammar. Object-oriented design metrics and tools a survey. INFOS'10.

[SM226] Brian Keith Miller, Dr. Pei Hsia, Dr. Chenho Kung. Objectoriented architecture measures. TSE'99.

[SM230] David Bellin, Manish Tyagi, Maurice Tyler. Objectoriented metrics an overview. CASCON'94.

[SM234] Bart Van Rompaey, Bart Du Bois, Serge Demeyer, Matthias Rieger. On The Detection of Test Smells A MetricsBased Approach for General Fixture and Eager Test. TSE'07.

[SM236] Eduardo Figueiredo, Claudio Sant'Anna, Alessandro Garcia, Thiago T. Bartolomei, Walter Cazzola, and Alessandro Marchetto. On the Maintainability of AspectOriented Software A ConcernOriented Measurement Framework. TSE'08.

[SM264] Michal Hocko and Tomas Kalibera. Reducing performance nondeterminism via cacheaware page allocation strategies. WOSP'10.

[SM266] Yoshiki Higo, Yoshihiro Matsumoto, Shinji Kusumoto, Katsuro Inoue. Refactoring Effect Estimation Based on Complexity Metrics. ASWEC'08.

[SM267] Mohammad Alshayeb. Refactoring Effect on Cohesion Metrics. ICC'09.

[SM269] Chandrashekar Rajaraman, Michael R. Lyu. Reliability and maintainability related software coupling metrics in C++ programs. TSE'92.

[SM275] Landry Chouambe, Benjamin Klatt, Klaus Krogmann. Reverse Engineering SoftwareModels of ComponentBased. TSE'08.

[SM277] Gyun Woo, Heung Seok Chae , Jian Feng Cui, JeongHoon Ji. Revising cohesion measures by considering the impact of write interactions between class members. INFSOF'08.

[SM288] Raghu V. Hudli, Curtis L. Hoskins, Anand V. Hudli. Software metrics for objectoriented designs. TSE'94.

[SM290] Marcio F. S.Oliveira, Ricardo Miotto Redin, Luigi Carro, Luis da Cunha Lamb, Flavio Rech Wagner. Software Quality Metrics and their Impact on Embedded Software. MOMPES'08.

[SM293] V. Lakshmi Narasimhan, B. Hendradjaya. Some theoretical considerations for a suite of metrics for the integration of software components. IS'07.

[SM301] Miro Casanova, Ragnhild Van Der Straeten, Viviane Jonckers. Supporting evolution in componentbased development using component libraries. CSMR'07.

[SM307] STEVE COUNSELL, STEPHEN SWIFT. The interpretation and utility of three cohesion metrics for objectoriented design. TOSEM'06.

[SM313] Robert C. Sharblet, Samuel S. Cohen. The objectoriented brewery a comparison of two objectoriented development methods. SIGSOFT'93.

[SM319] Reiner R. DiiInke,Ines Kuhrau. Toolbased quality management in objectoriented software development. TSE'94.

[SM320] Jan Wloka, Robert Hirschfeld, Joachim Hansel. Toolsupported Refactoring of Aspectoriented Programs. AOSD'08.

[SM323] Shmuel Rotenstreich. Toward measuring potential coupling. TSE'94.

[SM325] Letha Etzkom, Harry Delugach. Towards a semantic metrics suite for objectoriented design. TSE'00.

[SM327] Cristina Marinescu, Radu Marinescu, Tudor Grba. Towards a simplified implementation of objectoriented design metrics. METRICS'05.

[SM329] Thiago T. Bartolomei, Alessandro Garcia, Claudio SantAnna, Eduardo Figueiredo. Towards a Unified Coupling Framework for Measuring AspectOriented Programs. SOQUA'06.

[SM330] Avadhesh Kumar, Rajesh Kumar, P.S. Grover. Towards a Unified Framework for Cohesion Measurement in AspectOriented Systems. ASWEC'08.

[SM331] Avadhesh Kumar, Rajesh Kumar, P.S. Grover. Towards a Unified Framework for Complexity Measurement in AspectOriented Systems. CSSE'08.

[SM335] Jehad Al Dallal. Transitive based objectoriented lackofcohesion metric. WCIT'10.

[SM337] Moheb R. Girgis, Tarek. M. Mahmoud, Rehab R. Nour. UML class diagram metrics tool. TSE'09.

[SM341] Lionel C. Briand, Jurgen Wust, Hakim Lounis. Using coupling measurement for impact analysis in objectoriented systems. ICSM'99.

[SM343] Malcom Gethers, Denys Poshyvanyk. Using Relational Topic Models to capture coupling among classes in objectoriented software systems. ICSM'10.

[SM345] Andrea De Lucia, Rocco Oliveto, Luigi Vorraro. Using structural and semantic metrics to improve class cohesion. TSE'11.

[SM346] Meghan Revelle, Malcom Gethers, Denys Poshyvanyk. Using structural and textual information to capture feature coupling in objectoriented software. ESE'11.

[SM347] Mr. U. L. Kulkarni, Mr. Y. R. Kalshetty, Ms. Vrushali G. Arde. Validation of CK Metrics for Object-Oriented Design Measurement. ICETEC'10.

[SM350] Gunter Mussbacher, Daniel Amyot, Joao Araujo, Ana Moreira, Michael Weiss. Visualizing AspectOriented Goal Models with AoGRL. REV'07.