A Systematic Review on Mining Techniques for Crosscutting Concerns

Rafael S. Durelli
ICMC - USP
São Carlos, SP, Brazil
rdurelli@icmc.usp.br

Daniel S. M. Santibáñez
DC - UFSCar
São Carlos, SP, Brazil
daniel.santibanez@dc.ufscar.br

Nicolas Anquetil
RMoD Team - INRIA
Lille, France
Nicolas.Anquetil@inria.fr

Márcio E. Delamaro
ICMC - USP
São Carlos, SP, Brazil
delamaro@icmc.usp.br

Valter Vieira de Camargo
DC - UFSCar
São Carlos, SP, Brazil
valter@dc.ufscar.br

ABSTRACT

Background: The several maintenance tasks a system is submitted during its life usually cause its architecture deviates from the original conceivable design, ending up with scattered and tangled concerns across the software. The research area named concern mining attempts to identify such scattered and tangled concerns to support maintenance and reverse-engineering. Objectives: The aim of this paper is threefold: (*i*) identifying techniques employed in this research area, (*ii*) extending a taxonomy available on the literature and (*iii*) recommending an initial combination of some techniques. Results: We selected 62 papers by their mining technique. Among these papers, we identified 18 mining techniques for crosscutting concern. Based on these techniques, we have extended a taxonomy available in the literature, which can be used to position each new technique, and to compare it with the existing ones along relevant dimensions. As consequence, we present some combinations of these techniques taking into account high values of precision and recall that could improve the identification of both Persistence and Observer concerns. The combination that we recommend may serve as a roadmap to potential users of mining techniques for crosscutting concerns.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Miscellaneus

Keywords

Systematic Review, Concern Mining, Aspect Mining, Crosscutting Concerns.

1. INTRODUCTION

A possible definition for "software concern" is anything which stakeholders regard as a conceptual unit [11]. Examples of common software concerns include Persistence, Caching, Synchronization among others [5]. Developers and architects are continuously in need of up-to-date knowledge about the concerns currently implemented in their legacy system, and about the location of these concerns throughout the code. For example, during maintenance and reengineering, when there are bugs to be fixed, the maintenance task affects the whole implementation of a concern, and possibly to other concerns with which the fixed concern interacts.

Mining techniques for crosscutting concerns are indispensable for software maintenance, reverse engineering, reengineering and even for re-documentation [19]. However, manually applying a mining technique for crosscutting concern is difficult and error-prone. This came about because, legacy systems tend to: (*i*) have complex architectures with several clones spread out throughout the source code, (*ii*) involve several kinds of crosscutting concerns, e.g., patterns, architectural styles, business rules and non-functional properties and (*iii*) be very large, making the manual mining impractical. Thus, there is a need to use techniques and fully or semi-automated tools, which can aid software engineers to locate crosscutting concern into the legacy systems. In this context, the research area which aims to investigate techniques and tools to improve the mining of crosscutting concerns is known as "concern mining".

The aim of this paper is threefold. First, we aim to identify techniques employed in the research area herein described. Therefore, we have carried out a systematic review identifying mining techniques for crosscutting concerns. Second, we intent to extend the taxonomy presented by Kellens et al. [15]. Thus, we selected 62 papers and among them, we identified 18 mining techniques for crosscutting concerns. This taxonomy was proposed in 2007 and it needs to be updated because we found 7 new techniques developed in the past few years. Third, we recommend possible combinations of these techniques/tools that might improve recall and precision metrics for both Persistence and Observer concerns. We recommend four combinations of techniques discovered herein in order to improve recall and precision for Persistence concern. Similarly, we proposed two initial combinations of techniques identified to make better recall and precision metrics for Observer concern. This combination can be a motivation for potential users to improve the identification of well-known concerns.

This paper is organised as follows: In Section 2 presents how we have planned, conducted, reported and validated the systematic review. In addition, in this section there is also an extend taxonomy which was firstly proposed by Kellens et al. [15]. In Section 3 there are the threats to validity of our study. Section 4 presents a related work. Concluding remarks are made in Section 5.

2. THE SYSTEMATIC REVIEW

This study has been undertaken as a systematic review based on the guidelines proposed by Kitchenham and Brereton [16]. According to them, in order to conduct a systematic review, it is advisable to follow three main phases: (*i*) planning the review, (*ii*) conducting the review and (*iii*) reporting the review. Furthermore, in this paper we have used Visual Text Mining (VTM) technique to support the studies selection [18]. VTM uses text mining algorithms and methods combined with interactive visualisations. Therefore, it can help the user making sense of a collection of primary studies, without actually reading all of them. In this case the studies were reading partially or full. The following sections present details on how each phase was carried out.

2.1 Planning the Systematic Review

In this phase we have defined the review protocol. This protocol contains: (*i*) the research questions, (*ii*) the search strategy, (*iii*) the inclusion and exclusion criteria and (*iv*) the data extraction and synthesis method.

Research questions must embody the review study purpose. Moreover, these questions reflect the general scope of the review study. The scope is comprised of population (i.e., population group observed by the intervention), intervention (i.e., what is going to be observed in the context of the planned review study), and outcomes of relevance (i.e., the results of the intervention). Furthermore, during the conduction of this step, it was also necessary to establish the scope of the review study. According to the systematic review process [16], the scope has to be established using the PICO criteria. Thus, herein our **Population** is published scientific literature reporting on some existing mining technique for crosscutting concerns. The **Intervention** is published scientific literature interested with mining technique for crosscutting concerns. The **Comparison** is not applied herein. Finally, the **Outcomes of relevance** is an overview of the studies that have been conducted in the field of crosscutting concern mining, emphasizing primary studies that report on the techniques used in the research area, from observing such an aggregated data set, we also intend to provide insight into the frequencies of publication over time to inspect trends.

As described before, the objective of this review is to find out **which techniques are employed in mining techniques for crosscutting concerns. Moreover, we intent to extend the the taxonomy presented by Kellens et al. [15]. Finally, we also want to recommend possible combination of the identified techniques in order to identify remaining open research questions and possible avenues for future research**. In order to achieve such objectives we worked out five research questions. The questions are:

RQ$_1$: What are the mining techniques that are currently explored in the literature?

RQ$_2$: Which assessment techniques have been employed to evaluate these techniques and what are the results for common concerns?

RQ$_3$: Is there any difference in the precision and recall metrics when different techniques are used for mining the same concern?

RQ$_4$: Given a set of concerns, which are the most indicated techniques for performing the mining?

RQ$_5$: How can someone combine the techniques for improving the precision and recall metrics?

Afterwards, we have defined the search string and chosen the electronic databases. The search string was created based upon the following keywords: *approach, method, technique, methodology, aspect oriented, aspect-oriented, aspect mining, concern mining, coding mining, code mining, crosscutting concerns, cross-cutting concerns, Separation of Concern, SoC*. A sophisticated search string was constructed using boolean operators i.e., *AND*, *OR* and *NOT*. Figure 1 shows the search string elaborated. The search have encompassed electronic databases which are deemed as the most relevant scientific sources [9] and therefore likely to contain important primary studies. We have used the search string on the following electronic databases: *ACM* (portal.acm.org), *IEEE* (ieeexplore.ieee.org), *Scopus* (scopus.com) and *Springer* (springer.com/lncs).

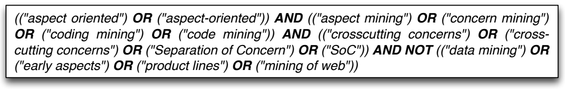
*(("aspect oriented") **OR** ("aspect-oriented")) **AND** (("aspect mining") **OR** ("concern mining") **OR** ("coding mining") **OR** ("code mining")) **AND** (("crosscutting concerns") **OR** ("cross-cutting concerns") **OR** ("Separation of Concern") **OR** ("SoC")) **AND NOT** (("data mining") **OR** ("early aspects") **OR** ("product lines") **OR** ("mining of web"))*

Figure 1: Search String.

Then, in order to determine which primary studies are relevant to answer our research questions, we have applied a set of inclusion and exclusion criteria. Inclusion criteria devised and applied are:

(a) **The primary study presents at least one mining technique for crosscutting concern:** the encountered techniques must assist the software engineer in the crosscutting concern mining.

(b) **The primary study presents at least one type of evaluation technique for mining techniques for crosscutting concern:** without the results of the evaluation we neither would be able to make comparisons desired nor would we propose a set of combination of the identified techniques.

Not all of these criteria must be present for every primary study. However, at least the former (a) must be present. If all criteria were mandatory, the number of selected techniques would decrease significantly.

Exclusion criteria devised and applied are:

(a) **The primary study presents data mining technique. Nevertheless, such technique is applied to databases and not for crosscutting concern mining:** techniques which are applied to databases were not included, since this sort of techniques is outside the scope of this paper.

(b) **The primary study is a short paper:** papers with two pages or less were not considered herein, since we considered that this kind of study do not own sufficient information.

We devised data extraction forms to accurately record the information obtained by the researchers from the primary studies. The form for data extraction provides some standard information, such as (*i*) name of the techniques identified, (*ii*) date of data extraction, (*iii*) title, authors, journal, publication details and (*iv*) a list of each conclusion and statement encountered for each sub-question.

During the extraction process, the data of each primary study were independently gathered by two reviewers. The review was performed in August, 2012 by a M.Sc. and a Ph.D. students; the achieved results were crossed and then validated. All the results of the search process are documented in the web material[1]. Therefore, it is clear to others how thorough the search was, and how they can find the same documents.

2.2 Conducting the Systematic Review

In this phase, firstly we identified primary studies in the digital libraries. The digital libraries Scopus has returned more primary studies than the others (262), i.e., IEEE, ACM and Springer have returned 215, 202 and 127, respectively. Possibly, this came about because this digital library indexes studies of others libraries, such as IEEE and Springer. Summing up, we have gotten 802 primary studies. Afterwards we have selected the primary studies by means of reading the titles and abstracts and the application of the inclusion and exclusion criteria. As a result, we have gotten a total of 124 primary studies that were read entirely, so the upshot obtained were 62 studies.

Among these 62 primary studies we have identified 18 mining techniques for crosscutting concern. Therefore, each included primary study was assigned to one or more techniques. In the following we outline each of the techniques:

- *Execution Patterns (EP)*: During program execution, program traces are generated, which reflect the run-time behavior of a software system. These traces are then investigated for recurring execution patterns. In [3] the authors introduce the notion of execution relations between method invocation. In other words, the basic idea of EP is to observer run-time behavior of software system and to extract information from the execution of the programs. As example, consider the Figure X, wherein that capitals represent methods names. According to Breu and Krinke [3] exist four different execution relations, they are: (*i*) outside-before, see Figure X, where B is called before A, (*ii*) outside-after, where A is called after B, see Figure X, (*iii*) inside-first, G is the first call in C and (*iv*) inside-last, H is the last call in C.

- *Dynamic Analysis (DA)*: DA is the analysis of the properties of a running program, whereas to static analysis, which examines a program's text to derive properties that hold for all executions, DA derives properties that hold for one or more executions by examination of the running program. While DA cannot prove that a program satisfies a particular property, it can detect violation of properties as well as provide useful information to programmers about the behavior of their programs. The usefulness of DA supplies from two of its essential characteristic:

  – Precision of information: DA typically involves instrumenting a program to examine or record certain aspects of its run-time state. This instrumentation can be tuned to collect precisely the information needed to address a particular problem. For example, to analyze the shape of data structures created by a program (lists, trees, dags, etc.), an instrumentation tool can be created to record the linkages among heap-allocated storage cells;

  – Dependence on program inputs: the very thing makes DA incomplete also supplies a powerful mechanism for relating program inputs and outputs to program behavior. With DA it is straightforward to relate changes in program inputs to changes in internal program behavior and program outputs, since all are directly observable and linked by the program execution. Viewed in this light, dynamic and static analysis might be better termed "input-centric" and "program-centric" analysis, respectively.

  Dynamic and static analyses are complementary techniques in a number of dimensions, them are:

  – Completeness: In general, dynamic analyses generate "dynamic program invariants", properties which are true for the observed set of execution. Static analysis can help determine or not these dynamic "invariants" truly are invariants over all program executions. In the cases where the dynamic and static analyses disagree, there are two possibilities: (*i*) the dynamic analysis is in error because it did not cover a sufficient number of executions, (*ii*) the static analysis is in error because it analyzed infeasible path;

  – Scope: Because dynamic analysis examines one very long program path, it has the potential to discover semantic dependencies between program entities widely separated in the path. Static analysis typically is restricted in the scope of a program it can analyze and efficiently, and may have troubles discovering such "dependencies at a distance".

  – Precision: DA has the benefit of examining the concrete domain of the program execution. In other hands, static analysis must abstract over this domain in order to ensure termination of the analysis, thus, it can lose information.

  In the context of discovering possible concerns DA applies FCA[2] to get execution traces. In other words, the obtained execution traces are analyzed using FCA for identifying methods and classes. Thus, methods

[1]http://tinyurl.com/99spmaz

[2]FCA is a mathematical theory of data analysis which describe relationship between a particular set of objects and a particular set of attributes [25]

belonging to more than one class may indicate presence of scattering code. If different methods from same class are specific to more than one use-case may indicate presence of tangling code [6].

- *Identifier Analysis (IA)*: This technique performs an identifier analysis using FCA algorithm. The assumption behind this approach is that relevant concerns in the source code are reflected by the use of naming conventions in the classes and methods of the system. As input to FCA algorithm, the classes and methods are used as objects and substrings generated from the classes and methods names are used as attributes. The resulting concepts consists out of maximal groups of classes and methods which share a maximal number of substrings [28].

- *Language Clues (LC)*: The approach uses natural language processing for mining crosscutting concern. The input is a collection of words from the source code and the output, chains of words which are semantically strongly related calculated with an algorithm. In order to mine for crosscutting concerns, they apply the chaining algorithm to the comments, method names, field names and class names of the system. A manual inspection to the resulting chains is needed in order to select possible concerns [27].

- *Method Clustering (MC)*: This technique starts by putting each method in a separate cluster and then, recursively, merges clusters by similarities in method names [1].

- *Call Clustering (CC)*: This technique starts from the assumption that if the same methods are called frequently from within different modules, then, they are closely related and must be clustered [32].

- *Fan-In analysis (FI)*: It is an approach that involves looking for methods that are called from many different call sites and whose functionality is needed across different methods, potentially spread over many classes and packages. FI is a semi-automated process consisting of three steps. Firstly, the method with the highest fan-in values need to be identified. Secondly, one have to filter out methods that may have a high fan-in but for which it is unlikely that there is a systematic pattern in their usage that could be exploited in an aspect solution. Common examples are getters and setters, and utility methods as well. Thirdly, the call sites of the high fan-in methods need to be inspected to determine if the method in questions does indeed implement crosscutting functionality. The last step is the most labor intensive, and it is based on an analysis of recurring patterns in, for example, the call sites of the high fan-in method [20]. High fan-in metric values may indicate presence of a crosscutting concern [32].

- *AST-Based Clone Detection (ACD)*: This technique takes the Abstract Syntax Tree (AST) of the source code into account. The output is a number of clone classes, i.e. groups of code fragments which are considered to be clones of each other [4].

- *Token-Based Clone Detection (TCD)*: This technique is based on lexical analysis of the source code. The output is a number of clone classes, i.e., groups of code fragments which are considered to be clones of each other [4].

- *History Based (HB)*: This technique intends to discover crosscutting concerns by analyzing the changes made in the source code along the time by using software repositories like revision control systems, files and databases [22].

- *Information Retrieval (IR)*: This technique tries to identify concerns. It is based on the similarity between terms used in the concern descriptions and in the program elements, e.g., element names, variable names. The results are ranked and a manual inspection is performed [10].

- *Parser-Based (PB)*: This technique performs a lexical or syntactic analysis of the source code to locate crosscutting concerns. It is based on the premise that code fragments which share concerns are likely to refer to readily identifiable shared entities such as identifiers and libraries [12].

- *PrefixSpan (PS)*: It is a data mining technique used to identify coding patterns in source code. Each method in a program is translated to a sequence that comprises method call elements and control elements. The algorithm searches for repetitive subsequences that could form a pattern [14].

- *Concern-Peers (CP)*: This technique identifies certain groups of code units that potentially share some crosscutting concerns. These code units, called concern peers, are detected based on their similar interactions (similar calling relations in similar contexts, either internally or externally). The algorithm scan for candidates, i.e., methods with similar code and names, then scan for peers and rank it to recommend possible concerns [23].

- *Method Call Tree (MCT)*: It uses method call tree to generate method call traces. These traces are then investigated for recurring method patterns based on different constraints, such as, the requirement that the patterns exist in always the same composition and in different calling contexts in the method call trace [26].

- *Data-Flow Concern Identification (DF)*: It is a semi-automated approach for concern identification specifically designed to support software understanding. It starts from a set of related variables and uses static dataflow information to determine the concern skeleton, a data-oriented abstraction of a concern [29].

- *Random Walks (RW)*: A random walk is a mathematical formalization of a trajectory that consists of taking successive random steps. This technique performs a random walks on the coupling graphs extracted from the program sources. The algorithm reflects the degrees of "popularity and significance" for each of the program elements on the coupling graphs. Filtering techniques, exploiting both types of ranks, are applied to produce a final list of candidate crosscutting concerns [31].

- *Model-Driven (MD)*: This technique is a model driven approach for concern mining and their separation, which automatically identifies desirable candidate concerns, without requiring input from the user. The concern miner acts as a model transformer converting the source code to a concern-oriented model [24].

The taxonomy proposed by Kellens et al. [15], takes into account 3 dimensions: *(i)* static or dynamic analysis; if the technique does a static analysis of the code or dynamic information which is obtained by executing the program or both. *(ii)* Token-Based or structural/behavioral analysis; lexical analysis like sequences of characters, regular expression or abstract syntax trees, type information, message sends, etc. *(iii)* Granularity: The level of granularity of the technique, method level or more fine-grained.

In this context, we have extended the taxonomy proposed by Kellens et al. [15] by means of adding the new identified mining techniques. In Figure 2, it is depicted our extended taxonomy. The small rectangles in the middle of the figure represent all of the techniques: the previous ones, proposed by Kellens et al. [15] and the new ones proposed by us, marked with an asterisk in Figure 2. More specifically, from the 18 techniques identified herein, 7 of them are new and were added to the taxonomy. The new techniques are: PrefixSpan, Information Retrieval, Dataflow, Model-Driven, Random Walks, History-Based, Concern Peers. The details on each of the identified new techniques and algorithms are outside the scope of this paper. Also, the fact of adding new techniques asked for the inclusion of new algorithms in the taxonomy as well. So, we added four algorithms, they are: Vector Space Indexing, Frecuent Itemset, Concern Model, Peer Detection. Finally, a new granularity level was added because the history-based technique can search for crosscutting concerns into source code repositories.

2.3 Validation

In validation phase an approach that uses VTM technique and the associated tool - Projection Explorer (PEx) - were applied to support the inclusion and exclusion decisions [18].

Figure 3 presents a document map generated using PEx. This map is composed of 802 primary studies analysed in this review, highlighting them using different shades of gray to differentiate in which of the stages a study was removed from the review. White points are studies excluded in first stage, gray points are the studies excluded in second stage and the black points are the included. The exploration of a document map is conducted in two steps: (*i*) firstly, a clustering algorithm is applied to the document map, creating groups of highly related documents; (*ii*) secondly, the resulting clusters are analysed in terms of: **Pure Clusters** - all documents belonging to a cluster have the same classification (all included or excluded, regardless of exclusion stage). Normally, in this case do not need to be reviewed; and **Mixed Clusters** - which represent documents with different classification on the same cluster. These cases are hints to the reviewer, and the estuaries grouped should be reviewed following the traditional method. To facility the visualisation, in Figure 3 just five clusters generated by PEx are depicted. Examples of pure clusters (all excluded) are identified in Figure 3 using label "(a)" and therefore do not needed to be reviewed. Mixed clusters (clusters containing black (included) and white or gray (excluded) studies) are identified using label "(b)" and they were reviewed by the

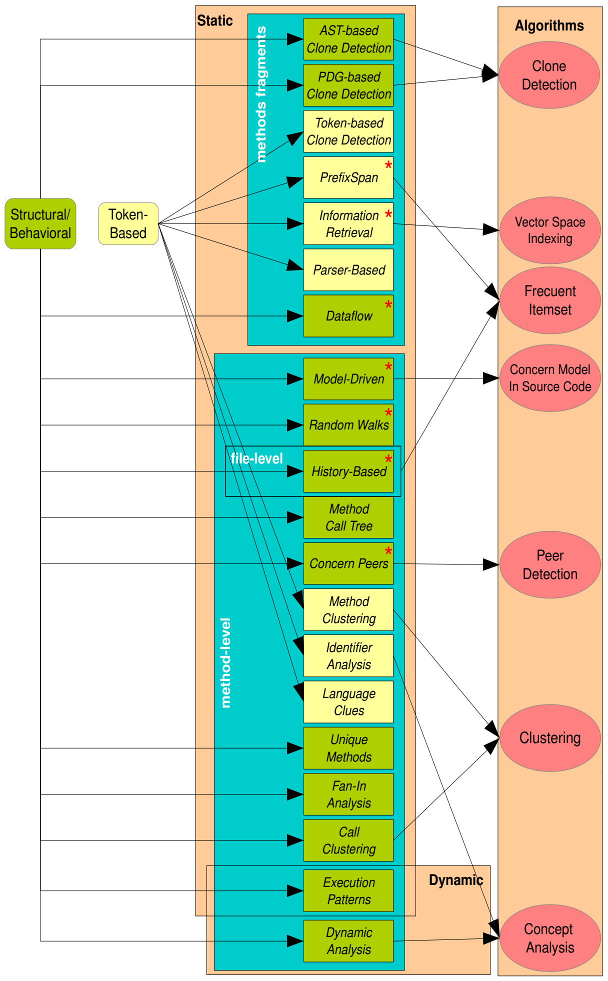

Figure 2: The Extended Taxonomy (Adapted from *Kellens et al.* [15]).

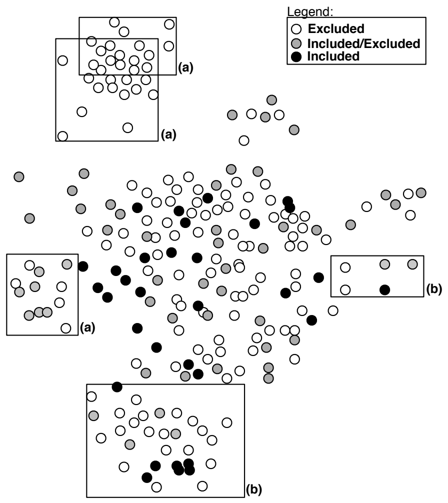

Figure 3: Document map colored with the history of the inclusions and exclusions of the studies.

authors of this paper. At the end, we kept the initial classifications conducted manually, but this technique contributed to a review of studies that could have been wrongly excluded or included previously.

2.4 Reporting the Systematic Review

The focus of this section is to present the broad overview of research within crosscutting concern mining we have acquired after classifying and categorizing primary studies. Moreover, we have used information drawn from this overview to answer this review study's research questions.

Aiming to show the frequencies of publication of all identified techniques for mining techniques for crosscutting concerns mining we have plotted a bubble plot, which is depicted in Figure 4. Bubble plots are essentially two x-y scatter plots with bubbles in category intersections. The size of each bubble is determined by the number of primary studies that have been classified as belonging to the categories corresponding to the bubble coordinates. This visual summary provides a bird's-eye view that enables one to pinpoint which categories have been emphasized in past research along with gaps and opportunities for future research.

In Figure 4 the facets we have used for organizing the map are the crosscutting concern mining techniques and year of publication. Based in this figure it is evident from observing it that we have found out 18 mining techniques for crosscutting concerns, as result we have answered the first part of the RQ_1. Based upon this bubble plot, we argue that the answer to second part of the RQ_1 is that Fan-In Analysis, Identifier Analysis and Dynamic Analysis are the techniques most used and Program Analysis Based, XScan-Concern-Peers, Data-Flow and Model Driven are the least used. More precisely, among the 62 primary studies included herein, 27 describe Fan-In Analysis, Identifier Analysis or Dynamic Analysis, respectively. In other hands, the techniques with less studies available in literature are Program Analysis Based, XScan-Concern-Peers, Data-Flow and Model Driven. Furthermore, it is argued that Fan-In Analysis, Identifier Analysis and Dynamic Analysis are evidence clusters (i.e., where there may be scope for more complete literature reviews to be undertaken). In contrast, Program Analysis Based, XScan-Concern-Peers, Data-Flow and Model Driven can be deemed as "evidence desert" (i.e., wherein better or new research is required).

The majority of selected primary studies are published by IEEE, i.e., 20 primary studies. The others primary studies have been selected from ACM, Scopus and Springer, 18, 16 and 8, respectively. As for the publication types, we have selected primaries studies that have been published in conferences, workshops and journals. The majority of the primary studies are conference paper (37), followed by workshop (16) and journal (9).

The way in which a technique is evaluated provides researchers and practitioners with useful information on the approach's quality, effectiveness, robustness, and practical applicability. Evaluating crosscutting concern mining techniques is difficult because defining the program elements that are relevant to a concern may be subjective. Despite this difficulty, researchers have devised some empirical strategies to assess them.

Empirical strategies of software engineering techniques are classified as experiment, case study and none [30]. In this context, we attempted to answer the RQ_2 by analyzing individually the 62 primary studies focus on gather which empirical strategies they have employed to validate the crosscutting concern mining techniques. In Figure 5 is depicted a pie chart wherein we have plotted the collected data.

As can be seen, among the 62 primary studies, 52 have carried out at least one empirical strategy. More specifically, 28 have carried out experiments to validate their crosscutting concern mining techniques and 24 primary studies have employed some sort of case studies. Only 10 primary studies neither have carried out experiments/case studies nor have made evident the use of specific evaluation strategies in order to validate their mining crosscutting concern techniques. Among the 52 primary studies, i.e., studies which carried out some evaluation techniques, we identified 31 studies using JHotDraw [2], a widely used framework to validate their mining crosscutting concern techniques. It is worth highlighting that from the universe of 31 studies using JHotDraw, 6 of them were evaluated by using two most well-known relevance-based measures of effectiveness, recall and precision. Table 1 presents the techniques and tools of these 6 studies with the precision and recall metrics for a particular JHotDraw version and where some crosscutting concerns were mentioned.

The data shown in Table 1 have useful information about how is the behavior in terms of precision and recall for different techniques regarding to JHotDraw concern. Note that there are some missing recall values because they were not reported. Recall is the proportion of relevant concern candidates that were discovered out of all concern candidates present in the source code. Thus a problem with calculating this metric, in a program under analysis, it is not known what the relevant concerns and code fragments are, except in an ideal case. In order to respond the RQ_3 take the persistence concern into account. It has a good precision value in most of the cases, that is, the percentage of relevant concern candidates in the set of all candidates reported was high. However, the recall value is uneven and this is a strong evidence that the universe of concern candidates used by each technique is not standardized. In other words, as shown in Table 1 usually there is a difference in the precision and recall metrics when different sorts of concerns are

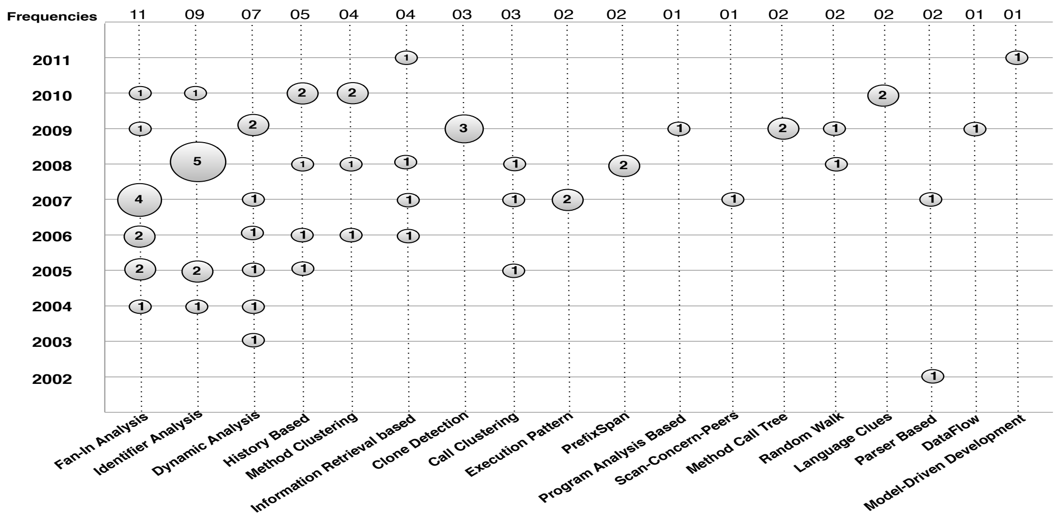

Figure 4: Map containing a year-wise distribution of publications on the all techniques found.

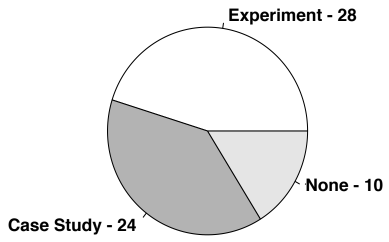

Figure 5: Which empirical strategies have been employed.

mined.

Based on the Table 1 we have answered the RQ$_4$. This table give us evidence that some techniques must be addressed to deal with certain kind of concerns instead of others. For instance, as shown in Table 1, the best techniques to mine Persistence concern are XScan and CBFA, as they have a precision of 100% - recall of 93% and precision of 80% - recall of 100%, respectively. Similarly, the best techniques to mine Iterator concern are CBFA and XScan, since the former has both Precision and Recall of 100% and the latter has a Precision of 100% and Recall of 89%.

In order to answer the RQ$_5$ we devised Table 2 and Table 3. Those tables show some candidate combinations of techniques described in Table 1. Combining these techniques can improve precision and recall metrics. We consider that someone could implement and/or reuse the best of several techniques to create a better mining technique for crosscutting concerns. For instance, as we have stated earlier, XScan is the best technique to mine Persistence concern, since it has a precision of 100%, but it has a recall of 93%. Therefore, maybe a solution to improve such recall, could be to combine other technique such as CFBA, which has a more reliable recall (100%) as shown in Table 1, i.e., first combination.

Also we stated that another solution to improve precision and recall metrics of techniques could be to combine the best techniques. For example, the second and the third combination illustrated in Table 2 represent this solution. We established that maybe the percentage of Dynamo will be improved if someone combine it with either CBFA or XScan. Finally, the last alternative is to combine Dynamo and MSAM, since according to the Table 1, the former has a good precision but it does not have a good recall, on the other side, the latter has a bad Precision but it has a good Recall. In the same way, Table 3, shows two candidate combinations in order to improve precision and recall for Observer concern.

Table 1: Precision and Recall for JHotDraw

Ref.	Tool	Precision	Recall	Concern	Version
Dynamic Analysis					
[7]	Dynamo	64% 100%	49% 26%	Undo Persistence	5.4
Concern Peers					
[23]	XScan	90% 100% 100% 97% 100%	93% 89% 93% 100% 100%	Undo Iterator Persistence Observer Visitor	6
Method Clustering + Fan-In					
[23]	CBFA	100% 100% 80% 86% 86%	86% 100% 100% 80% 100%	Undo Iterator Persistence Observer Visitor	6
Information Retrieval					
[13]	MSAM	5%	100%	Undo Persistence Observer Command	-
Method Clustering					
[8]	Clustering Algorithms	87.5%	-	Observer *Consistent Behaviour* *Contract Enforcement* Command	5.2
Call Clustering					
[17]	SOM	51%	-	Observer *Consistent Behaviour* *Contract Enforcement* Command	5.4
Fan-In					
[21]	FINT	30%	-	*Consistent Behavior* *Contract Enforcement*	5.4

*The values for MSAM were calculated using two thresholds, 0.4 for precision and 0.5 for recall.

3. THREATS TO VALIDITY

Table 2: Combination for Persistence

N	Combined Technique
1st	CFBA + XScan
2nd	Dynamo + CBFA
3rd	Dynamo + XScan
4th	MSAM + Dynamo

Table 3: Combination for Observer

N	Combined Technique
1st	MSAM + CBFA
2nd	MSAM + Clustering Algorithms

Primary studies selection. Aiming at ensuring an unbiased selection process, we defined research questions in advance and devised inclusion and exclusion criteria we believe are detailed enough to provide an assessment of how the final set of primary studies was obtained. However, we cannot rule out threats from a quality assessment perspective, we simply selected studies without assigning any scores. In addition, we wanted to be as inclusive as possible, thus no limits were placed on date of publication and we avoided imposing many restrictions on primary study selection since we wanted a broad overview of the research area.

Missing important primary studies. The search for primary studies was conducted in several search engines, even though it is rather possible we have missed some primary studies. Nevertheless, this threat was mitigated by selecting search engines which have been regarded as the most relevant scientific sources [9] and therefore prone to contain the majority of the important studies.

Reviewers reliability. All the reviewers of this study are researchers in the software reuse field, focused on the aspect-oriented programming, software testing and software product line, and none of the techniques and tools developed by us. Therefore, we are not aware of any bias we may have introduced during the analyses.

Data extraction. Another threat for this review refers to how the data were extracted from the digital libraries, since not all the information was obvious to answer the questions and some data had to be interpreted. Therefore, in order to ensure the validity, multiple sources of data were analyzed, i.e. papers, technical reports, white papers. Furthermore, in the event of a disagreement between the two primary reviewers, a third reviewer acted as an arbitrator to ensure full agreement was reached.

4. RELATED WORK

Closely related work to this review is a survey with aspect mining techniques [15], which details and compares a large selection of automated techniques and aspect mining tools. The goal is to present a comparative framework for distinguishing aspect mining techniques, and assess known techniques against this framework.

5. CONCLUDING REMARKS

In this paper we presented a systematic review of mining techniques for crosscutting concern, following the process described by Kitchenham [9]. Through a examination of 62 primary studies encompassing techniques to mine crosscutting concern, this review has presented 18 techniques. Researchers can use this review as a basis for advancing the field, while practitioners can use it to identify techniques that are well-suited to their needs. This systematic review should serve not only academic researchers but also industrial professionals, aiming at adopting some techniques to mine crosscutting concern within their organizations. The review described in this paper reveals that the most mentioned mining techniques for crosscutting concern are Fan-In Analysis, Identifier Analysis and Dynamic Analysis. In contrast, Program Analysis Based, XScan-Concern-Peers, Data-Flow and Model Driven can be deemed as "evidence desert".

Based on the identified techniques we have extended the taxonomy proposed by Kellens et al. [15]. This new taxonomy contains 7 new mining techniques for crosscutting concerns. By using this taxonomy we hold that this taxonomy could serve as an initial roadmap to crosscutting concern researchers. Moreover, this extended taxonomy could be relevant for tool developers who might have knowledge about the best aspect indicators to use or who may have certain demands about the granularity of the results.

The main future directions that emerged from this review are the need for empirical, comparative evaluations and the opportunity for developing combined techniques. Indeed, since every technique relies on different assumptions and uses different underlying analysis techniques, the found techniques are highly complementary, which suggests the possibility of several useful combinations. Thus, through the results obtained in this review we argue that if one pretends to devise a new mining techniques for crosscutting concerns to mine either Persistence or Observer, a good initial point is to take into consideration the combination herein illustrated in Table 2 and 3 but more studies are needed because the combinations proposed did not take into consideration the versions of the system, so we intend to analyze this in future works.

6. ACKNOWLEDGMENTS

Rafael Durelli would like to thank the financial support provided by FAPESP, process number 2012/05168-4. Daniel Santibáñez would like to thank the financial support provide by CAPES. Valter Camargo would like to thank CNPQ, process number 560241/2010-0. We also thank Thiago Gottardi for his proofreading of this paper.

7. REFERENCES

[1] M. L. Bernardi and G. A. Di Lucca. Conan: A tool for the identification of crosscutting concerns. In *16th Working Conference on Reverse Engineering*, Washington, DC, USA, 2009. IEEE Computer Society.

[2] J. Brant. *HotDraw*. Masters thesis, 1995.

[3] S. Breu and J. Krinke. Aspect mining using event traces. In *Proceedings of the 19th IEEE international conference on Automated software engineering*, Washington, DC, USA, 2004. IEEE Computer Society.

[4] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwe. On the use of clone detection for identifying crosscutting concern code. *IEEE Trans. Softw. Eng.*, 31:804–818, October 2005.

[5] V. Camargo, R. Ramos, and P. Masiero. Implementacao de variabilidades em frameworks orientados a aspectos desenvolvidos em aspectj. *1st Brazilian Workshop on Aspect-Oriented Software Development (WASP)*, 2004.

[6] M. Ceccato. Automatic support for the migration towards aspects. In *Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering*, pages 298–301, Washington, DC, USA, 2008. IEEE Computer Society.

[7] M. Ceccato and P. Tonella. Dynamic aspect mining. *Software, IET*, 3(4):321 –336, august 2009.

[8] G. Cojocar and G. Czibula. On clustering based aspect mining. In *Intelligent Computer Communication and Processing, 2008. ICCP 2008. 4th International Conference on*, pages 129 –136, aug. 2008.

[9] T. Dyba, T. Dingsoyr, and G. K. Hanssen. Applying systematic reviews to diverse study types. In *International Symposium on Empirical Software Engineering and Measurement*, ESEM '07, pages 225–234, Washington, DC, USA, 2007. IEEE Computer Society.

[10] M. Eaddy, A. V. Aho, G. Antoniol, and Y.-G. Guéhéneuc. Cerberus: Tracing requirements to source code using information retrieval, dynamic analysis, and program analysis. In *Proceedings of the 2008 The 16th IEEE International Conference on Program Comprehension*, pages 53–62, Washington, DC, USA, 2008. IEEE Computer Society.

[11] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho. Do crosscutting concerns cause defects? *IEEE Trans. Softw. Eng.*, 34:497–515, July 2008.

[12] W. G. Griswold, Y. K. Y, and J. J. Yuan. Aspect browser: Tool support for managing dispersed aspects. In *Separation of Concerns in Object-oriented Systems*, 1999.

[13] J. Huang, Y. Lu, and J. Yang. Aspect mining using link analysis. In *5th International Conference on Frontier of Computer Science and Technology*, pages 312–317, Washington, DC, USA, 2010. IEEE Computer Society.

[14] T. Ishio, H. Date, T. Miyake, and K. Inoue. Mining coding patterns to detect crosscutting concerns in java programs. In *15th Working Conference on Reverse Engineering*, pages 123–132, Washington, DC, USA, 2008. IEEE Computer Society.

[15] A. Kellens, K. Mens, and P. Tonella. Transactions on aspect-oriented software development iv. chapter A survey of automated code-level aspect mining techniques, pages 143–162. Berlin, Heidelberg, 2007.

[16] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering. *Information and Software Technology*, 51:7–15, January 2009.

[17] S. G. Maisikeli. *Aspect mining using self-organizing maps with method level dynamic software metrics as input vectors.* PhD thesis, 2010.

[18] V. Malheiros, E. Hohn, R. Pinho, M. Mendonca, and J. C. Maldonado. A visual text mining approach for systematic reviews. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, ESEM '07, pages 245–254, Washington, DC, USA, 2007. IEEE Computer Society.

[19] M. Marin, A. V. Deursen, and L. Moonen. Identifying crosscutting concerns using fan-in analysis. *ACM Trans. Softw. Eng. Methodol.*, 17:3:1–3:37, December 2007.

[20] M. Marin, A. V. Deursen, and L. Moonen. Identifying crosscutting concerns using fan-in analysis. *ACM Trans. Softw. Eng. Methodol.*, 17:1–37, 2007.

[21] M. Marin, L. Moonen, and A. van Deursen. A common framework for aspect mining based on crosscutting concern sorts. In *13th Working Conference on Reverse Engineering*, pages 29–38, Washington, DC, USA, 2006. IEEE Computer Society.

[22] F. Mulder and A. Zaidman. Identifying cross-cutting concerns using software repository mining. In *International Workshop on Principles of Software Evolution*, pages 23–32, New York, NY, USA, 2010. ACM.

[23] T. T. Nguyen, H. V. Nguyen, H. A. Nguyen, and T. N. Nguyen. Aspect recommendation for evolving software. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 361–370, New York, NY, USA, 2011. ACM.

[24] B. Nora and S. Ghoul. A model-driven approach to aspect mining. In *27th International Conference on Software Engineering*, pages 361–370, New York, NY, USA, 2006. ACM.

[25] U. Priss. Formal concept analysis in information science. *Annual Rev. Info. Sci & Technol.*, pages 521–543, 2006.

[26] L. Qu and D. Liu. Aspect mining using method call tree. In *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering*, pages 407–412, Washington, DC, USA, 2007. IEEE Computer Society.

[27] D. Shepherd, L. Pollock, and T. Tourwé. Using language clues to discover crosscutting concerns. In *Proceedings of the 2005 workshop on Modeling and analysis of concerns in software*, pages 1–6, New York, NY, USA, 2005. ACM.

[28] T. Tourwe and K. Mens. Mining aspectual views using formal concept analysis. In *Source Code Analysis and Manipulation, Fourth IEEE International Workshop*, pages 97–106, Washington, DC, USA, 2004. IEEE Computer Society.

[29] M. Trifu. Using dataflow information for concern identification in object-oriented software systems. In *Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering*, pages 193–202, Washington, DC, USA, 2008. IEEE Computer Society.

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

[31] C. Zhang and H.-A. Jacobsen. Mining crosscutting concerns through random walks. *IEEE Transactions on Software Engineering*, 2011.

[32] D. Zhang, Y. Guo, and X. Chen. Automated aspect recommendation through clustering-based fan-in analysis. In *Automated Software Engineering, 2008.*, pages 278 –287, 2008.