On Some Criteria for Comparing Aspect Mining Techniques

Grigoreta Sofia Cojocar
Department of Computer Science
Babeş-Bolyai University
1, Mihail Kogălniceanu Street
Cluj-Napoca, Romania
grigo@cs.ubbcluj.ro

Gabriela Şerban
Department of Computer Science
Babeş-Bolyai University
1, Mihail Kogălniceanu Street
Cluj-Napoca, Romania
gabis@cs.ubbcluj.ro

ABSTRACT

Many aspect mining techniques have already been proposed, even if aspect mining is a relatively new research domain. That is why the necessity to classify and to compare them has emerged. Not all the characteristics of the aspect mining techniques were considered as comparison criteria and very few generally applicable evaluation measures were proposed. This paper proposes a set of new criteria to compare the existing aspect mining techniques and a set of new evaluation measures to compare the results obtained by these techniques. The applicability of these measures for different aspect mining techniques is also discussed.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Restructuring, reverse engineering, and reengineering*; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Measurement, Performance

Keywords

aspect mining, evaluation measures

1. INTRODUCTION

Aspect Mining (AM) is a relatively new research domain that tries to identify crosscutting concerns in legacy software system. Nevertheless, many aspect mining techniques have already been proposed, and the necessity to classify and to compare them has emerged. The existing AM techniques differ in many ways: the approach used (formal concept analysis based, metrics based, clustering based, etc.), the development lifecycle when they can be applied, the type of analysis used (static or dynamic), the granularity level, etc., making it difficult to compare and to classify.

Workshop LATE '07 March 12-13, 2007 Vancouver, British Columbia, Canada

Very few comparisons between AM techniques were done so far. The first comparison is presented in [7] where three AM techniques (Fan-in [17], Identifier analysis [22] and Execution traces [21]) are compared. The type of crosscutting concerns discovered for JHotDraw ([10]) case study and the possibility of combination with other AM techniques are used as comparison criteria. No classification considering the obtained results is given.

In [8], the results obtained by the above mentioned AM techniques, and combinations of them, for JHotDraw case study are further analyzed. A detailed discussion of the obtained results is presented, using *recalled methods* and *seed quality* as evaluation measures.

Paper [11] presents a comparison of code level AM techniques. Among the criteria used for comparison are: the type of data analyzed by an AM technique, the granularity, the user involvement, the preconditions. No classification considering the results obtained on a case-study is given.

In [18] a small comparison using the crosscutting concern's symptoms is presented. The authors also consider as comparison criterion whether the aspect mining tool requires or not user involvement. No classification using the results obtained on a case study is given.

In this paper we present a set of new comparison criteria that are important when considering the use of an AM technique or when trying to classify AM techniques, and a set of new evaluation measures to compare the results obtained by AM techniques. These are the main contributions of this paper.

The paper is structured as follows. In Section 2 a set of new comparison criteria for AM techniques is introduced. New evaluation measures, and their applicability to six different AM techniques, are defined in Section 3. Conclusions and further work are given in Section 4.

2. COMPARISON CRITERIA

In this section we propose a set of new comparison criteria and we briefly discuss their importance when comparing AM techniques. In our view, input data required by the aspect mining technique, availability, language dependency, and measures are important criteria that must be strongly considered when thinking about using a particular AM technique or when thinking about comparing the results obtained by different AM techniques.

Input data required

This is an important criterion when considering to use a particular AM technique. It indicates to a new user what

data he/she must have in order to be able to use the AM technique. Most techniques require as input the source code of the system to be mined ([2, 17, 21, 22]), but there are a few techniques like *History Mining* ([3]) and *EA-Miner* ([19]) that require a different input (CVS transactions, text documents, etc.). We mention that for AM techniques that use dynamic analysis, we consider as input data the source code, because the system must be modified in order to obtain the necessary information during execution. The traces needed by the dynamic analysis cannot always be obtained without modifying the source code of the system.

Language dependency

The theoretical part of each AM technique is independent on the language used (programming or natural language). However, the tool associated with the technique is language dependent, restricting the use of the technique to systems that were developed using the same language. For example, *FINT* ([16]) or *Dynamo* ([9]) can be used only for Java based systems (considering the current version of the tool). *WMATRIX* ([23]) for EA-Miner ([19]) is dependent on English. It cannot be used for a system whose requirements are written in French or Italian.

Availability

The availability of the tool/results/case study are important issues for a new user of an AM technique and/or for someone that tries to compare AM techniques.

Most of the existing AM techniques have an associated tool. If an AM technique does not have an associated tool or the tool is not publicly available, it is very unlikely that people will use the technique, as they would first need to build a tool. The tool availability is also important when trying to compare the results obtained by different AM techniques on new case studies. Even though most AM techniques have an associated tool, few tools are publicly available (*FINT* for Fanin, *Dynamo* for Execution traces and *WMATRIX* for EA-Miner).

The availability of the results obtained by an AM technique on a particular case study is also important, if the associated tool is not available. The developers of other AM techniques can use them to evaluate and to compare their techniques. The results obtained by different AM techniques on the same case study are very important when trying to build the reference set of crosscutting concerns for the considered case study, especially when different crosscutting concerns are discovered by different AM techniques for the same case study. As can be seen in Table 1, the results obtained for different case studies are publicly available for very few AM techniques.

The case study availability is important when considering to compare different AM techniques based on the obtained results. If a case study used by an AM technique is not publicly available, it cannot be used for comparison with other techniques.

Evaluation Measures

Measures are essential when considering to compare AM techniques based on the obtained results. Some measures have been used to evaluate the results obtained by AM techniques [6, 8, 15]. In most cases, the measures used are applicable only to those particular AM techniques. So far, only two measures are generally applicable (the number of false positives and the number of false negatives). However, in order to compare different AM techniques, based on the obtained results, evaluation measures are needed. These measures should be applicable for as many AM techniques as possible. That is why in Section 3 we propose a set of new evaluation measures to compare the results obtained by AM techniques. They are generally defined, and their applicability to six AM techniques is also discussed.

3. A SET OF NEW EVALUATION MEASURES

Based on the analysis of the existing AM techniques, we have observed that, besides the criteria proposed in [11, 18], they also differ in the following aspects:

- The choice of the system representation.
- The way the elements from the system are considered candidate seeds.
- The contexts chosen to determine crosscuttingness.

In Subsection 3.1 we present the theoretical model on which we base our approach. A set of new measures for evaluating the results of different AM techniques is presented in Subsection 3.2.

3.1 Theoretical Model

Let us consider a software system S as a pair of two sets E and CTX ($S =< E, CTX >$). $E = \{e_1, e_2, \dots e_s\}$ is the set of all the elements that exist in the system S. Depending on the aspect mining technique, an element can be a method, a statement, a word, etc. $CTX = \{ctx_1, ctx_2, \dots, ctx_q\}$ represents the contexts in which the elements from E appear in the system S. A context can be a class, a method, a sentence, etc.

We also consider a crosscutting concern CC as a pair of two sets EL_{CC} and CT_{CC} ($CC =< EL_{CC}, CT_{CC} >$).
$EL_{CC} = \{el_1, el_2, \dots, el_{s_{CC}}\} \subset E$ represents the elements that constitute the crosscutting concern CC.
$CT_{CC} = \{ctt_1, ctt_2, \dots ctt_{q_{CC}}\} \subset CTX$ represents the different contexts in which the elements from EL_{CC} appear as part of the crosscutting concern CC.
We denote by $TCCC$ the set of all crosscutting concerns present in the system S ($TCCC = \{CC_1, CC_2, \dots, CC_r\}$).

Considering the above, in our view, an aspect mining technique $\mathcal{T}$ is a pair of two functions $Select$ and $Context$ ($\mathcal{T} =< Select, Context >$).
The $Select$ function gives the elements from E that belong to crosscutting concerns, i.e. $Select(S) = SE \subseteq E$.
The $Context$ function gives the contexts from CTX in which the elements given by $Select$ appear ($Context : SE \rightarrow \mathcal{P}(CTX)$, $Context(e) = SCT_e \subseteq CTX$). We have denoted by $\mathcal{P}(CTX)$ the set of all subsets from CTX. For a given element $e \in SE$, this function returns contexts from CTX where the element e appears (all or only a part of them, depending on the aspect mining technique).

The two sets, E and CTX, and the two functions, $Select$ and $Context$, are particular to each aspect mining technique (see Subsection 3.3).

3.2 Definitions

Based on the theoretical model introduced in Subsection 3.1 we define in this subsection a set of new measures that can be used for evaluating the results obtained by different AM techniques. We will use the notations from Subsection 3.1, and two additional ones:

- *Res* denotes the result obtained by the *Select* function, $Res = Select(E)$.
- $ContextS(A)$ denotes the set of contexts in which the elements from A appear. If $A = \{a_1, a_2, \ldots, a_t\} \subset E$, then $ContextS(A) = \bigcup_{a \in A} Context(a)$.

We define below new evaluation measures for comparing the results obtained by AM techniques. Comparing to the existing evaluation measures, the measures proposed in this subsection are general and distinguish between the elements of a crosscutting concern and the contexts in which they appear. Such a distinction was not reported before in the general case. There is only one measure (*seed-quality*) introduced in [15], that refers to a particular AM technique.

DEFINITION 1. ***Elements Coverage - ElemCov***
ElemCov *represents the percentage of the elements discovered by Select that do belong to the crosscutting concern $CC \in TCCC$. It is formally defined as:*

$$ElemCov(CC, Res) = \frac{|EL_{CC} \cap Res|}{|EL_{CC}|}.$$

DEFINITION 2. ***Missed Elements - MissedElem***
MissedElem *represents the percentage of elements that do belong to CC, but that were not chosen by Select. It is formally defined as:*

$$MissedElem(CC, Res) = \frac{|EL_{CC} \setminus Res|}{|EL_{CC}|}.$$

We mention that the following equation holds:
$ElemCov(CC, Res) + MissedElem(CC, Res) = 1$.

DEFINITION 3. ***Wrong Elements - WrongElem***
WrongElem *represents the percentage of elements chosen by Select, that do not belong to any crosscutting concerns from $TCCC$. It is formally defined as:*

$$WrongElem(TCCC, Res) = \frac{|Res \setminus \bigcup_{CC \in TCCC} EL_{CC}|}{|Res|}.$$

As most of the existing aspect mining techniques do not group the results based on concerns, **WrongElem** measure is defined using the total set of crosscutting concerns $TCCC$ from the system S. Most of the existing AM techniques call this measure *false positives*.

The above defined measures are based on *precision/recall* measures used in information retrieval systems ([14]).

DEFINITION 4. ***Context Coverage - ContextCov***
ContextCov *represents the percentage of the contexts discovered by Context which actually belong to the crosscutting concern $CC \in TCCC$. It is formally defined as:*

$$ContextCov(CC, Res) = \frac{|CT_{CC} \cap ContextS(EL_{CC} \cap Res)|}{|CT_{CC}|}.$$

DEFINITION 5. ***Missed Context - MissedContext***
MissedContext *represents the percentage of contexts that were not discovered by Context as belonging to CC, but that actually do belong to the crosscutting concern $CC \in TCCC$. It is formally defined as:*

$$MissedContext(CC, Res) = \frac{|CT_{CC} \setminus ContextS(EL_{CC} \cap Res)|}{|CT_{CC}|}.$$

DEFINITION 6. ***Wrong Context - WrongContext***
WrongContext *represents the percentage of contexts discovered by Context, that do not belong to the crosscutting concern $CC \in TCCC$. It is formally defined as:*

$$WrongContext(CC, Res) = \frac{|ContextS(EL_{CC} \cap Res) \setminus CT_{CC}|}{|ContextS(EL_{CC} \cap Res)|}.$$

An average measure can be defined for all the measures proposed above (except **WrongElem**), that gives the average percentage for each measure, considering all the crosscutting concerns present in the system S. For example, for **ElemCov**, the average measure, called **AverageElemCov** is defined as follows:

$$AverageElemCov(TCCC, Res) = \frac{1}{r} \sum_{i=1}^{r} ElemCov(CC_i, Res),$$

where $r = |TCCC|$.

The **AverageMissedElem** measure represents the average percentage of *false negatives*.

We mention that it can be proved that the values of all the above defined measures are between $[0, 1]$. For **ElemCov** and **ContextCov** measures larger values are desirable (the ideal value is 1). For **WrongElem**, **MissedElem**, **WrongContext**, and **MissedContext** measures smaller values are desirable (the ideal value is 0).

The above defined measures should be simultaneously considered in order to obtain a more accurate evaluation of an AM technique.

We discuss, in the following, some situations that might appear for the values of the above defined measures, and their possible causes.

1. The value of **ElemCov** is high, and the value of **MissedContext** is also high (possibly higher than the value of **ContextCov**). In this case the *Context* function must be revised.
2. The value of **WrongElem** is high (e.g., higher that 0.5). The *Select* function must be improved.
3. The value of **ContextCov** is high, and the value of **WrongContext** is approximately the same. The *Context* function must be improved.

3.3 Applicability

In this subsection we present how the measures proposed above can be applied to six different aspect mining techniques: EA-Miner ([19, 20]), Clone detection ([5, 6]), Fan-in ([17]), Execution relations ([1, 2, 12, 13]), Execution traces ([21]), and History Mining ([3, 4]).

- **EA-Miner**
  The *elements* of the software system are the words from the requirements documents, interviews, etc. The *contexts* considered for each word are the sentences where it appears.
  The *Select* function chooses the words having the same part of speech in many sentences. The *Context* function for a given word, w, returns the sentences in which w appears and it has the same part of speech.

- **Clone detection**
  The *elements* of the software system are small sequences of statements from the source code. The *contexts* considered are the parts of the source code files where

these sequences appear.
The *Select* function chooses those sequences of statements that appear approximately the same in different parts of the source code, depending on the clone detection tool used. The *Context* function for a given sequence *seq* of statements returns the parts from the source code where *seq* appears, as considered by the clone detection tool used.

- **Fan-in**
The *elements* of the software system are the methods that appear in the source code. The *contexts* are the methods that call them in the source code.
The *Select* function chooses all the methods whose number of callers is greater than a given threshold. The *Context* function for a given method m returns all the methods that call m.

- **Execution relations**
The *elements* are the methods from the source code. The *contexts* are the execution relations that exist between the methods. The relations may be obtained from the program trace ([1, 2])) or from the source code ([12, 13])).
The AM techniques based on execution relations return as results four sets of pairs ([2]), of the form $u \circ v$, that satisfy the uniformity and crosscutting constraints, where $\circ$ represents the type of the execution relation. The methods chosen by *Select* are from the left-side of the relation, and the *Context* for a method m, are the methods from the right-side of the corresponding relations.
For example, if $\circ$ is an execution relation and the corresponding result is $R^{\circ} = \{u_1 \circ v_a, u_1 \circ v_b, u_1 \circ v_c, u_2 \circ v_d, u_2 \circ v_e\}$, the methods chosen by *Select* are $\{u_1, u_2\}$, and $Context(u_1) = \{v_a, v_b, v_c\}$ and $Context(u_2) = \{v_d, v_e\}$.

- **Execution traces** The *elements* are the methods from the source code. The *contexts* are the use-cases that use these methods in their implementation.
The *Select* function chooses the methods that are used in the implementation of at least two use-cases. The *Context* function for a given method m returns the use-cases where m was used.

- **History Mining**
The *elements* are the methods from the source code. The *contexts* are the CVS transactions corresponding to add-method-call operations.
The *Select* function chooses the methods that have at least 8 add-method-call operations, meaning that a call to these methods was added at least 8 times. The *Context* function for a given method m returns the methods (locations) where the calls to m were added.

3.4 Evaluation

The measures defined in Subsection 3.2 are general enough in order to evaluate most aspect mining techniques. We have not made an evaluation, yet, for the following reasons:

1. There is no case study publicly available, for which the complete set of existing crosscutting concerns is also publicly available.
2. In order to compute the values of the above proposed measures, we need the tool associated to the AM technique or the results obtained by the technique must be publicly available. Only a few tools are publicly available, and even less results obtained by AM techniques for a certain case study (see Table 1).
3. The case study used for comparison should have (publicly) available all the data required by existing AM techniques. Even if most of the techniques require as input the source code, there are a few that require different input. For example, JHotDraw cannot be used for *History Mining* as there is no CVS archive available.

However, we intend to compare a part of the existing AM techniques using JHotDraw as case study. We will use the union of the sets of crosscutting concerns, publicly available, that were identified by different AM techniques as the reference set of existing crosscutting concerns.

4. CONCLUSIONS AND FURTHER WORK

We have presented a set of new criteria that should be considered when comparing aspect mining techniques. We have also defined a set of new evaluation measures to compare the results obtained by different aspect mining techniques. We did not used them, yet, but we intend to compute the proposed measures for a part of AM techniques, using JHotDraw as case study.

Although the measures proposed in this paper are generally applicable, there are still some open issues that we will further investigate:

- Have the values for a certain measure the same significance for AM techniques that are applicable at different levels of granularity (statement, method)? i.e., for **ElemCov**, is a percentage of 0.7 for statement based techniques the same with 0.7 for method based techniques?
- Can AM techniques that are applicable at different stages from the development lifecycle be compared? Suppose that there is a software system for which the requirements documents, the design and the source code are available. If we denote by CCC_{Req} the set of crosscutting concerns discovered during requirements, by CCC_{Design} the set discovered during design and by CCC_{Code} the set discovered from the source code, intuitively the following relation should hold $CCC_{Req} \subseteq CCC_{Design} \subseteq CCC_{Code}$. Is this true in practice?

5. REFERENCES

[1] S. Breu. Aspect Mining Using Event Traces. Master's thesis, University of Passau, Germany, March 2004.

[2] S. Breu and J. Krinke. Aspect Mining using Event Traces. In *Proceedings of International Conference on Automated Software Engineering*, pages 310–315, 2004.

[3] S. Breu and T. Zimmermann. Mining Aspects from History. In *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, pages 221–230, September 2006.

[4] S. Breu, T. Zimmermann, and C. Lindig. Mining Eclipse for Crosscutting Concerns. In *Proceedings of the 2006 International Workshop on Mining Software Repositories (MSR '06)*, pages 94–97, New York, NY, USA, 2006. ACM Press.

Technique	Case study	Case-study language	Case study availability	Results availability
EA-Miner	Auction System	English	Yes	No
	Light Control System	English	Yes	No
	Library System	English	No	No
Clone detection	CC part of ASML	C	No	No
Fan-in	JHotDraw v5.4b1	Java	Yes	Yes
	Petstore	Java	Yes	Yes
	TomCat	Java	Yes	Yes
Event Traces	Graffiti (Gravisto)	Java	Yes	No
	AspectJ Telecom	Java	Yes	Yes
	Anchovis	n/a	No	No
	JHotDraw v5.4b1	Java	Yes	No
Execution traces	Carla Laffra, Dijkstra's algorithm	Java	Yes	Yes
	JHotDraw v5.4b1	Java	Yes	Yes
History Mining	Eclipse v3.2M3	Java	Yes	No

Table 1: Some of the case studies used in aspect mining

[5] M. Bruntink, A. van Deursen, T. Tourwé, and R. van Engelen. An Evaluation of Clone Detection Techniques for Identifying Crosscutting Concerns. In *ICSM '04: Proceedings of the 20th IEEE International Conference on Software Maintenance*, pages 200–209, Washington, DC, USA, 2004. IEEE Computer Society.

[6] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwé. On the use of clone detection for identifying crosscutting concern code. *IEEE Transactions on Software Engineering*, 31(10):804–818, 2005.

[7] M. Ceccato, M. Marin, K. Mens, L. Moonen, P. Tonella, and T. Tourwé. A Qualitative Comparison of Three Aspect Mining Techniques. In *IWPC '05: Proceedings of the 13th International Workshop on Program Comprehension*, pages 13–22. IEEE Computer Society, 2005.

[8] M. Ceccato, M. Marin, K. Mens, L. Moonen, P. Tonella, and T. Tourwé. Applying and Combining Three Different Aspect Mining Techniques. *Software Quality Control*, 14(3):209–231, 2006.

[9] Dynamo - Dynamic Aspect Mining Tool. http://star.itc.it/dynamo/.

[10] E. Gamma. JHotDraw Project. http://sourceforge.net/projects/jhotdraw.

[11] A. Kellens, K. Mens, and P. Tonella. A Survey of Automated Code-level Aspect Mining Techniques. *Transactions on Aspect-Oriented Software Development, Special Issue on Software Evolution*, 2007, to appear.

[12] J. Krinke. Mining Control Flow Graphs for Crosscutting Concerns. In *Proceedings of the 13th Working Conference on Reverse Engineering (WCRE 2006)*, pages 334–342, Washington, DC, USA, 2006. IEEE Computer Society.

[13] J. Krinke and S. Breu. Control-Flow-Graph-Based Aspect Mining. In *Workshop on Aspect Reverse Engineering (WARE)*, 2004.

[14] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[15] M. Marin, L. Moonen, and A. van Deursen. A common framework for aspect mining based on crosscutting concern sorts. In *WCRE '06: Proceedings of the 13th Working Conference on Reverse Engineering (WCRE 2006)*, pages 29–38, Washington, DC, USA, 2006. IEEE Computer Society.

[16] M. Marin, L. Moonen, and A. van Deursen. FINT: Tool Support for Aspect Mining. In *WCRE '06: Proceedings of the 13th Working Conference on Reverse Engineering (WCRE 2006)*, pages 299–300, Washington, DC, USA, 2006. IEEE Computer Society.

[17] M. Marin, A. van, Deursen, and L. Moonen. Identifying Aspects Using Fan-in Analysis. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004)*, pages 132–141. IEEE Computer Society, 2004.

[18] B. Nora, G. Said, and A. Fadila. A comparative classification of aspect mining approaches. *Journal of Computer Science*, 2(4):322–325, 2006.

[19] A. Sampaio, R. Chitchyan, A. Rashid, and P. Rayson. Ea-miner: a Tool for Automating Aspect-Oriented Requirements Identification. In *ASE '05: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 352–355, New York, NY, USA, 2005. ACM Press.

[20] A. Sampaio, N. Loughran, A. Rashid, and P. Rayson. Mining Aspects in Requirements. In *Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design Workshop (held with AOSD 2005)*, Chicago, Illinois, USA, 2005.

[21] P. Tonella and M. Ceccato. Aspect Mining through the Formal Concept Analysis of Execution Traces. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 112–121, Washington, DC, USA, 2004. IEEE Computer Society.

[22] T. Tourwé and K. Mens. Mining Aspectual Views using Formal Concept Analysis. In *Proc. IEEE International Workshop on Source Code Analysis and Manipulation*, 2004.

[23] Wmatrix corpus analysis and comparison tool. http://www.comp.lancs.ac.uk/ucrel/wmatrix/.