A Formal Model For Clustering Based Aspect Mining

GRIGORETA SOFIA MOLDOVAN
Babeş-Bolyai University
Department of Computer Science
1, M. Kogălniceanu Street, Cluj-Napoca
ROMANIA

GABRIELA ŞERBAN
Babeş-Bolyai University
Department of Computer Science
1, M. Kogălniceanu Street, Cluj-Napoca
ROMANIA

Abstract: Aspect mining is a research direction that tries to identify crosscutting concerns in already developed software systems, without using Aspect Oriented Programming. The aim of this paper is to propose a new formal model for clustering based aspect mining. Such a formal model was not defined in the literature, yet. We also define quality measures for evaluating the results of clustering based aspect mining techniques. A small example on how to compute these measures is also provided.

Key–Words: aspect mining, clustering, formal model, quality measures.

1 Introduction

The Aspect Oriented Programming (AOP) is a new paradigm that is used to design and implement *crosscutting concerns* [4]. A *crosscutting concern* is a feature of a software system that is spread all over the system, and whose implementation is tangled with other features' implementation. Logging, persistence, and connection pooling are well-known examples of crosscutting concerns. Some of the benefits that the use of AOP brings to software engineering are: better modularization, higher productivity, software systems that are easier to maintain and evolve.

Aspect mining is a relatively new research direction that tries to identify crosscutting concerns in already developed software systems, without using AOP. The goal is to identify them and then to refactor them to aspects, to achieve a system that can be easily understood, maintained and modified.

Many aspect mining techniques have been proposed so far ([1], [2], [5], [6], [8], [9], [10]). Some of them are dynamic ([1], [2], [10]), some of them are static ([5], [6], [9]), some of them use *formal concept analysis* ([10]), some of them use *fan-in analysis* ([5], [6]), some of them use *clustering* ([2], [6], [9]) and some use *clone detection techniques* ([8]).

All the aspect mining techniques have the following characteristics:

- They use a representation of the mined software system.
- The software system is divided into groups.
- These groups (or a part of them) are analyzed in order to identify crosscutting concerns.

In this paper we are focusing on clustering based aspect mining techniques. In the clustering-based aspect mining techniques proposed so far ([2], [6], [9]) a software system is considered as a set of methods that are grouped in classes (clusters) using clustering techniques ([3]). A part of these clusters are then analyzed in order to discover crosscutting concerns.

In [7] a set of quality measures for evaluating the results of clustering based aspect mining techniques were proposed. The theoretical model on which the measures were defined was a restrictive one, because a software system was considered as a set of methods.

In this paper we propose a formal model for clustering based aspect mining. As far as we know, such a model was not proposed in the literature, yet. This model is a generalization of the one presented in [7]. All the theoretical aspects on which this model was developed are also provided.

The paper is structured as follows. In Section 2 we introduce a formal model for clustering based aspect mining. The quality measures for evaluating the results are defined in Section 3. Section 4 presents a small example on how to compute the quality measures. Some conclusions and further work are given in Section 5.

2 Formal Model

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a software system, where $s_i, 1 \leq i \leq n$ is an element from the system. An *element* can be a statement, a method, a class, a module, etc. We denote by n ($|S|$) the number of elements of the system.

In the following, we will consider a crosscutting concern as a set of elements $C \subset S$, $C = \{c_1, c_2, ..., c_{cn}\}$, elements that implement this concern. The number of elements in the crosscutting concern C is $cn = |C|$. Let $CCC = \{C_1, C_2, ..., C_q\}$ be the set of all crosscutting concerns that exist in the system S. The number of crosscutting concerns in the system S is $q = |CCC|$. Let $NCCC = S - (\bigcup_{i=1}^{q} C_i)$ be the set of elements from the system S, elements that are not used to implement any crosscutting concerns.

Definition 1 ***Partition of a system*** S**.**
The set $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *is called a* ***partition*** *of the system* S *iff* $1 \leq p \leq n$, $K_i \subseteq S, K_i \neq \emptyset, \forall\, i, 1 \leq i \leq p$, $S = \bigcup_{i=1}^{p} K_i$ *and* $K_i \cap K_j = \emptyset$, $\forall\, i, j, 1 \leq i, j \leq p, i \neq j$.

In the following we will refer to K_i as the i-th *cluster* of $\mathcal{K}$ and to $\mathcal{K}$ as a *set of clusters*.

Formally, the problem of aspect mining can be viewed as the problem of identifying a partition $\mathcal{K}$ of the software system S.

A partition of a software system S can be obtained by an aspect mining technique, in our case a clustering based aspect mining one.

Abstractly, a clustering based aspect mining technique $\mathcal{T}$ can be viewed as a pair of functions:

$$\mathcal{T} = (divide, analyze).$$

$divide$ is a function that maps a software system S to a partition $\mathcal{K}$ of the system S, i.e., $divide(S) = \mathcal{K}$. $analyze$ is a function that indicates the clusters from $\mathcal{K}$ that will be analyzed by the user of the technique, i.e., $analyze(S, \mathcal{K}) = \mathcal{SK}, \mathcal{SK} \subset \mathcal{K}$.

In order for a technique $\mathcal{T}$ to be efficient, equality (1) should hold:

$$CCC = \mathcal{SK}. \tag{1}$$

In practice, equality (1) is hard to be satisfied, that is why, it is acceptable that $CCC \subseteq \mathcal{SK}$. However, as smaller the set $\mathcal{SK} \setminus CCC$ is, as efficient $\mathcal{T}$ is.

Definitions 2 and 3 will give optimality conditions for the result of a clustering based aspect mining technique.

Definition 2 ***Good partition of a system*** S**.**
Being given a partition $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *of the system* S, $\mathcal{K}$ *is called a* ***good partition*** *of the system* S *with respect to the set* $CCC = \{C_1, C_2, ..., C_q\}$ *of all crosscutting concerns, iff:*
(1) $p \geq q$
(2) *there exists a permutation* $\{\sigma(1), \sigma(2), ..., \sigma(q)\}$ *of the set* {1,2, ..., q} *such that* $C_{\sigma(i)} \subseteq K_i, \forall\, i, 1 \leq i \leq q$.

Intuitively, $\mathcal{K}$ is a good partition of the system S if all the elements implementing a crosscuting concern C_i $(1 \leq i \leq q)$ are in the same cluster K_{j_i} $(1 \leq j_i \leq p)$.

Definition 3 ***Optimal partition of a system*** S**.**
Being given a partition $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ *of the system* S, $\mathcal{K}$ *is called an* ***optimal partition*** *of the system* S *with respect to the set* $CCC = \{C_1, C_2, ..., C_q\}$ *of all crosscutting concerns, iff:*
(1) $p \geq q$
(2) *there exists a permutation* $\{\sigma(1), \sigma(2), ..., \sigma(q)\}$ *of the set* {1,2, ..., q} *such that* $K_i = C_{\sigma(i)}, \forall\, i, 1 \leq i \leq q$.

Intuitively, $\mathcal{K}$ is an optimal partition of the system S if all the elements implementing a crosscuting concern C_i $(1 \leq i \leq q)$ are in the same cluster K_{j_i} $(1 \leq j_i \leq p)$ and they are the only elements in K_{j_i}.

Remark 4 *An optimal partition of a software system* S *is a good partition where the elements implementing a crosscutting concern are the only elements in the corresponding cluster.*

3 Quality Measures

In this section we propose a set of new quality measures that will be used to determine if a partition $\mathcal{K}$ of a software system S is optimal with respect to a set of crosscutting concerns CCC.

The notations described in Section 2 will be also used in this section.

Definition 5 ***DISPersion of crosscutting concerns - DISP.***
The dispersion of the set CCC *in the partition* $\mathcal{K}$, *denoted by* $DISP(CCC, \mathcal{K})$, *is defined as*

$$DISP(CCC, \mathcal{K}) = \frac{1}{|CCC|} \sum_{i=1}^{|CCC|} disp(C_i, \mathcal{K}). \tag{2}$$

$disp(C, \mathcal{K})$ *is the dispersion of a crosscutting concern* C *and is defined as:*

$$disp(C, \mathcal{K}) = \frac{1}{|D_C|}, \tag{3}$$

where

$$D_C = \{k | k \in \mathcal{K} \text{ and } k \cap C \neq \emptyset\}. \quad (4)$$

D_C is the set of clusters that contain elements which are also in C.

In our view, $DISP(CCC, \mathcal{K})$ defines the dispersion degree of crosscutting concerns in clusters. For a crosscutting concern C, $disp(C, \mathcal{K})$ indicates the number of clusters that contain elements belonging to C.

Lemma 6 *If $\mathcal{K}$ is a partition of the software system S and CCC is the set of crosscutting concerns in S, then inequality (5) holds:*

$$0 < DISP(CCC, \mathcal{K}) \leq 1. \quad (5)$$

Proof: Because $\forall C_i \in CCC, C_i \subset S$, and $\mathcal{K}$ is a partition of S, there must be at least one cluster $K_{C_i} \in \mathcal{K}$ such that $C_i \cap K_{C_i} \neq \emptyset$. It follows that:

$$D_{C_i} \neq \emptyset \quad (6)$$

From (6) and the definition of D_{C_i} (4), we have:

$$\emptyset \subset D_{C_i} \subseteq \mathcal{K} \quad (7)$$

From (7) it follows that:

$$1 \leq |D_{C_i}| \leq |\mathcal{K}|, \ \forall C_i \in CCC \quad (8)$$

From (8) and (3) we have:

$$\sum_{i=1}^{|CCC|} \frac{1}{|\mathcal{K}|} \leq \sum_{i=1}^{|CCC|} disp(C_i, \mathcal{K}) \leq \sum_{i=1}^{|CCC|} 1$$

$\Rightarrow$

$$\frac{1}{|\mathcal{K}|} \leq DISP(CCC, \mathcal{K}) \leq 1 \quad (9)$$

Inequality (9) implies (5), so Lemma 6 is proved. □

Remark 7 *Larger values for $DISP$ indicate better partitions with respect to CCC, meaning that $DISP$ has to be maximized.*

Definition 8 ***DIVersity of a partition - DIV.***
The diversity of a partition $\mathcal{K}$ with respect to the set CCC, denoted by $DIV(CCC, \mathcal{K})$, is defined as

$$DIV(CCC, \mathcal{K}) = \frac{1}{|\mathcal{K}|} \sum_{i=1}^{|\mathcal{K}|} div(CCC, K_i). \quad (10)$$

$div(CCC, k)$ is the diversity of a cluster $k \in \mathcal{K}$ and is defined as:

$$div(CCC, k) = \frac{1}{|V_k| + \tau(k)} \quad (11)$$

where

$$V_k = \{C | C \in CCC \text{ and } k \cap C \neq \emptyset\} \quad (12)$$

is the set of crosscutting concerns that have elements in k, and

$$\tau(k) = \begin{cases} 1 & \text{if } k \cap NCCC \neq \emptyset \\ 0 & \text{if } k \cap NCCC = \emptyset \end{cases} \quad (13)$$

.

$\tau(k)$ is 1 if the cluster k contains elements that do not implement any crosscutting concern, and 0 otherwise.

$DIV(CCC, \mathcal{K})$ defines the degree to which each cluster contains elements from different crosscutting concerns or elements from other concerns.

Lemma 9 *If $\mathcal{K}$ is a partition of the software system S and CCC is the set of crosscutting concerns in S, then inequality (14) holds:*

$$0 < DIV(CCC, \mathcal{K}) \leq 1. \quad (14)$$

Proof: From (12) it follows that:

$$0 \leq |V_k| \leq |CCC| \quad (15)$$

Using (15) and the definition of $\tau(k)$ (13), we have:

$$0 \leq |V_k| + \tau(k) \leq |CCC| + 1 \quad (16)$$

$|V_k| + \tau(k)$ is a non-zero value, as:

(i) if $|V_k| = 0 \overset{k \neq \emptyset}{\Rightarrow} \tau(k) = 1$;

(ii) if $\tau(k) = 0 \overset{k \neq \emptyset}{\Rightarrow} V_k \neq \emptyset \Rightarrow |V_k| \geq 1$.

From (i) and (ii) it follows that:

$$1 \leq |V_k| + \tau(k) \leq |CCC| + 1, \ \forall k \in \mathcal{K} \quad (17)$$

From (17) and (11) we have:

$$\sum_{i=1}^{|\mathcal{K}|} \frac{1}{|CCC| + 1} \leq \sum_{i=1}^{|\mathcal{K}|} div(CCC, K_i) \leq \sum_{i=1}^{|\mathcal{K}|} 1$$

$\Rightarrow$

$$\frac{1}{|CCC| + 1} \leq DIV(CCC, \mathcal{K}) \leq 1 \quad (18)$$

Inequality (18) implies (14), so Lemma 9 is proved. □

Remark 10 *Larger values for DIV indicate better partitions with respect to CCC, meaning that DIV has to be maximized.*

Based on Lemmas 6 and 9, we will define Theorem 11 that gives the conditions for a partition of a software system to be optimal with respect to its set of crosscutting concerns.

Theorem 11 *If $\mathcal{K}$ is a partition of the software system S and CCC is the set of crosscutting concerns in S, then $\mathcal{K}$ is an **optimal partition** (Definition 3) iff $DISP(CCC, \mathcal{K}) = 1$ and $DIV(CCC, \mathcal{K}) = 1$.*

Proof: First, we will prove implication "$\Rightarrow$" from Theorem 11.

If $\mathcal{K}$ is an optimal partition, from Definition 3 we have that:

(i)
$$\forall C \in CCC,\ |D_C| = 1$$
$$\Rightarrow$$
$$DISP(CCC, \mathcal{K}) = 1.$$

(ii)
$$\forall k \in \mathcal{K},\ |V_k| + \tau(k) = 1.$$

Based on Definition 3, there are two possibilities: $|V_k| = 1$ and $\tau(k) = 0$ or $|V_k| = 0$ and $\tau(k) = 1$ $\Rightarrow DIV(CCC, \mathcal{K}) = 1$.

So, implication "$\Rightarrow$" from Theorem 11 is proved.

Now, we will prove implication "$\Leftarrow$" from Theorem 11.

If $DISP(CCC, \mathcal{K}) = 1$ and $DIV(CCC, \mathcal{K}) = 1$, from Equations (2) and (10) we have that:

(i)
$$\forall C_i \in CCC,\ disp(C_i, \mathcal{K}) = 1 \Rightarrow |D_{C_i}| = 1.$$

$\Rightarrow \mathcal{K}$ is a good partition (Definition 2).

(ii)
$$\forall k \in \mathcal{K},\ div(CCC, k) = 1 \Rightarrow |V_k| + \tau(k) = 1$$

There are two possibilities:

1. $|V_k| = 1$ and $\tau(k) = 0 \Rightarrow \exists C_i \in CCC$ s.t. $k = C_i \Rightarrow k \in CCC$;
2. $|V_k| = 0$ and $\tau(k) = 1 \Rightarrow \neg\exists C_i \in CCC$ s.t. $k \cap C_i \neq \emptyset \Rightarrow k \subseteq NCCC$.

From (i) and (ii) it follows that $\mathcal{K}$ is an optimal partition (Definition 3).

So, implication "$\Leftarrow$" is proved and Theorem 11 is also proved. □.

In the following, we will give in Definitions 12 and 20 measures for evaluating the efficiency of a clustering based aspect mining technique.

Definition 12 ***Percentage of ANalyzed Elements for a partition - PANE.***
Let us consider that the partition $\mathcal{K}$ is analyzed in the following order: $K_1, K_2, ..., K_p$.

The percentage of analyzed elements for a partition K with respect to the set CCC, denoted by $PANE(CCC, \mathcal{K})$, is defined as:

$$PANE(CCC, \mathcal{K}) = \frac{1}{|CCC|} \sum_{i=1}^{|CCC|} pane(C_i, \mathcal{K}).$$

$pane(C, \mathcal{K})$ is the percentage of the elements that need to be analyzed in the partition $\mathcal{K}$ in order to discover the crosscutting concern C, and is defined as:

$$pane(C, \mathcal{K}) = \frac{1}{|S|} \sum_{j=1}^{r} |K_j|$$

where $r = max\{t \,|\, 1 \leq t \leq p$ and $K_t \cap C \neq \emptyset\}$ is the index of the last cluster in the partition $\mathcal{K}$ that contains elements from C.

$PANE(CCC, \mathcal{K})$ defines the percentage of the number of elements that need to be analyzed in the partition in order to discover all crosscutting concerns that are in the system S. We consider that a crosscutting concern was discovered when all the elements that implement it were analyzed.

Lemma 13 *If $\mathcal{K}$ is a partition of the software system S and CCC is the set of crosscutting concerns in S, then inequality (19) holds:*

$$0 < PANE(CCC, \mathcal{K}) \leq 1. \tag{19}$$

The proof of Lemma 13 is similar to the proofs of Lemmas 6 and 9.

Remark 14 *Smaller values for $PANE$ indicate shorter time for analysis, meaning that $PANE$ has to be minimized.*

Definition 15 ***PRECision of a clustering based aspect mining technique - PREC.***
Let $\mathcal{T}$ be a clustering based aspect mining technique.

The precision of $\mathcal{T}$ with respect to a partition $\mathcal{K}$ and the set CCC, denoted by $PREC(CCC, \mathcal{K}, \mathcal{T})$, is defined as:

$$PREC(CCC, \mathcal{K}, \mathcal{T}) = \frac{1}{|CCC|} \sum_{i=1}^{|CCC|} prec(C_i, \mathcal{K}, \mathcal{T}).$$

$$prec(C_i, \mathcal{K}, \mathcal{T}) = \begin{cases} 1 & \text{if } C_i \text{ was discovered by } \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$$

is the precision of $\mathcal{T}$ with respect to the crosscutting concern C_i.

$PREC(CCC, \mathcal{K}, \mathcal{T})$ defines the percentage of crosscutting concerns that are discovered by $\mathcal{T}$. In all clustering based aspect mining techniques, only a part of the clusters are analyzed, meaning that some crosscutting concerns may be missed.

Lemma 16 *If $\mathcal{K}$ is a partition of the software system S, CCC is the set of crosscutting concerns in S, and $\mathcal{T}$ is a clustering based aspect mining technique, then inequality (20) holds:*

$$0 \leq PREC(CCC, \mathcal{K}, \mathcal{T}) \leq 1. \qquad (20)$$

The proof of Lemma 16 is similar to the proofs of Lemmas 6 and 9.

Remark 17 *Larger values for $PREC$ indicate better partitions with respect to CCC, meaning that $PREC$ has to be maximized.*

If a partition $\mathcal{K}$ of a software system S was obtained, the order in which the clusters from $\mathcal{K}$ are analyzed influence the values of $PANE$ and $PREC$ measures.

That is why, Definitions 18 and 19 establish **efficient** partitions from the aspect mining point of view.

Definition 18 ***Ideal partition of a system S.***
*Being given a partition $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ of the system S, $\mathcal{K}$ is called an **ideal partition** of the system S with respect to the set $CCC = \{C_1, C_2, ..., C_q\}$ of all crosscutting concerns if $\mathcal{K}$ is an **optimal partition** (Definition 3) with $PREC$ equal to 1 and a minimum value for $PANE$.*

Definition 19 ***Almost ideal partition of a system S.***
*Being given a partition $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ of the system S, $\mathcal{K}$ is called an **almost ideal partition** of the system S with respect to the set $CCC = \{C_1, C_2, ..., C_q\}$ of all crosscutting concerns if $\mathcal{K}$ is a **good partition** (Definition 2) with $PREC$ equal to 1 and a minimum value for $PANE$.*

4 Example

In the following, a small example showing how to compute $DISP$, DIV, and $PANE$ measures is presented. $PREC$ is not discussed because it depends on the clustering based aspect mining technique.

Let $S = \{s_1, s_2, ..., s_{17}\}$ be a software system with 17 elements, and let $C_1 = \{s_2, s_3, s_{17}\}$ and $C_2 = \{s_1, s_4, s_8\}$ be the crosscutting concerns that exist in the system S ($CCC = \{C_1, C_2\}$).

Let $\mathcal{K} = \{K_1, K_2, K_3, K_4, K_5\}$ be a partition of the software system S, where:

$K_1 = \{s_2, s_{16}\}$
$K_2 = \{s_3, s_7, s_8, s_9, s_{17}\}$
$K_3 = \{s_1, s_4, s_5, s_{12}\}$
$K_4 = \{s_6, s_{10}, s_{13}\}$
$K_5 = \{s_{11}, s_{14}, s_{15}\}$

DISP

Using Definition 5, we have to compute $disp(C, \mathcal{K})$ for each $C \in CCC$. The obtained values are shown below.

Crosscutting concern	**The set D**	***disp***
C_1	$\{K_1, K_2\}$	0.5
C_2	$\{K_2, K_3\}$	0.5

Based on the definition, $DISP(CCC, \mathcal{K}) = 0.5$.

DIV

Using Definition 8, we have to compute $div(CCC, k)$ for each $k \in \mathcal{K}$. The obtained values are shown below.

Cluster	**The set V**	τ	***div***
K_1	$\{C_1\}$	1	0.5
K_2	$\{C_1, C_2\}$	1	0.33
K_3	$\{C_2\}$	1	0.5
K_4	$\emptyset$	1	1
K_5	$\emptyset$	1	1

Based on the definition, $DIV(CCC, \mathcal{K}) = 0.66$.

PANE

Using Definition 12, we have to compute $pane(C, \mathcal{K})$ for each $C \in CCC$. First, we have to determine the index of the last analyzed cluster that contains element(s) from C. The obtained values are shown below.

Crosscutting concern	**# last cluster**	***pane***
C_1	2	0.41
C_2	3	0.64

Based on the definition, $PANE(CCC, \mathcal{K}) = 0.52$.

For our example, an **optimal partition** (Definition 3) is:

$K_1 = \{s_6, s_9, s_{10}\}$
$K_2 = \{s_2, s_3, s_{17}\}$
$K_3 = \{s_{11}, s_{12}, s_{14}, s_{15}\}$
$K_4 = \{s_5, s_7, s_{13}, s_{16}\}$
$K_5 = \{s_1, s_4, s_8\}$

and an **ideal partition** (Definition 18) is:

$K_1 = \{s_2, s_3, s_{17}\}$
$K_2 = \{s_1, s_4, s_8\}$
$K_3 = \{s_{11}, s_{12}, s_{14}, s_{15}\}$
$K_4 = \{s_5, s_7, s_{13}, s_{16}\}$
$K_5 = \{s_6, s_9, s_{10}\}$

5 Conclusions and Further Work

In this paper we have proposed a new formal model for clustering based aspect mining. A set of new quality measures for evaluating the results of clustering based aspect mining techniques were defined.

Using this measures we have given conditions for the optimality and efficiency of the results obtained by clustering based aspect mining techniques.

Although the formal model proposed in this paper is for clustering based aspect mining, it is general enough in order to be applied for other aspect mining techniques (like the one presented in [5]).

As a conclusion, even if Aspect Mining can be considered (as we have shown in Section 2) a practical software engineering problem, this paper has presented a formal model of it.

Further work can be done in the following directions:

- To generalize the formal model and the quality measures for other aspect mining techniques (like [1], [5], [10]).
- To identify other possible quality measures for evaluating clustering approaches in aspect mining.
- To determine quality measures that can be applied for evaluating all aspect mining techniques (including clustering approaches).

References:

[1] S. Breu and J. Krinke. Aspect Mining Using Event Traces. In *Proc. International Conference on Automated Software Engineering (ASE)*, pages 310–315, 2004.

[2] L. He and H. Bai. Aspect Mining using Clustering and Association Rule Method. *International Journal of Computer Science and Network Security*, 6(2A):247–251, February 2006.

[3] A. Jain, M. N. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[4] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *Proceedings European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, 1997.

[5] 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.

[6] G. S. Moldovan and G. Serban. Aspect Mining using a Vector-Space Model Based Clustering Approach. In *Proceedings of Linking Aspect Technology and Evolution Workshop(LATE 2006)*, March 2006.

[7] G. S. Moldovan and G. Serban. Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques. In *Proceedings of Towards Evaluation of Aspect Mining(TEAM) Workshop, ECOOP*, pages 13–16, 2006.

[8] Orlando Alejo Mendez Morales. Aspect Mining Using Clone Detection. Master's thesis, Delft University of Technology, The Netherlands, August 2004.

[9] D. Shepherd and L. Pollock. Interfaces, Aspects, and Views. In *Proceedings of Linking Aspect Technology and Evolution Workshop(LATE 2005)*, March 2005.

[10] P. Tonella and M. Ceccato. Aspect Mining through the Formal Concept Analysis of Execution Traces. In *Proceedings of the IEEE Eleventh Working Conference on Reverse Engineering (WCRE 2004)*, pages 112–121, November 2004.