A Partitional Clustering Algorithm for Crosscutting Concerns Identification

GABRIELA CZIBULA
Babeș-Bolyai University
Department of Computer Science
1, M. Kogălniceanu Street
Cluj-Napoca, ROMANIA
gabis@cs.ubbcluj.ro

GRIGORETA SOFIA COJOCAR
Babeș-Bolyai University
Department of Computer Science
1, M. Kogălniceanu Street
Cluj-Napoca, ROMANIA
grigo@cs.ubbcluj.ro

ISTVÁN GERGELY CZIBULA
Babeș-Bolyai University
Department of Computer Science
1, M. Kogălniceanu Street
Cluj-Napoca, ROMANIA
istvanc@cs.ubbcluj.ro

Abstract: Identifying crosscutting concerns is an important issue in the maintenance of software systems. It aims at refactoring the existing systems to use aspect oriented programming, in order to make them easier to maintain and to evolve. In this paper we present a new partitional clustering algorithm for identifying crosscutting concerns in existing software systems. We experimentally evaluate our algorithm using the open source case study JHotDraw, providing a comparison of the proposed approach with similar existing approaches.

Key–Words: Aspect mining, crosscutting concerns, clustering

1 Introduction

Nowadays, software systems have become more and more complex and large. Usually, a software system is composed of many core concerns and (some) crosscutting concerns (like logging, exception handling). If core concerns can be cleanly separated and implemented using existing programming paradigms, this is not true for crosscutting concerns, as a crosscutting concern has a more system-wide behavior that cuts across many of the core concerns implementation modules. Separation of concerns [17] is a very important principle of software engineering that, in its most general form, refers to the ability to identify, encapsulate and manipulate those parts of software that are relevant to a particular concept, goal, or purpose. The aspect oriented paradigm (AOP) is one of the approaches proposed, so far, for designing and implementing crosscutting concerns [10]. Aspect oriented techniques allow crosscutting concerns to be implemented in a new kind of module called aspect, by introducing new language constructs like pointcuts and advices.

Aspect mining is a 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.

There exists many reasons for migrating a legacy system to an aspect oriented based system. An inadequate solution for crosscutting concerns implementation has a negative impact on the final system with consequences like duplicated code, scattering of concerns throughout the entire system and tangling of concern-specific code with that of other concerns. These consequences lead to software systems that are hard to maintain and to evolve. When aspect oriented techniques are used, the crosscutting concerns are cleanly separated from the core concerns.

The main contribution of this paper is to introduce a partitional clustering algorithm for identifying crosscutting concerns in existing software systems.

The rest of the paper is structured as follows. Section 2 presents some existing work in the field of aspect clustering. The main aspects related to the problem of clustering, particularly to partitional clustering are presented in Section 3. A new partitional clustering algorithm (PACO) for identifying crosscutting concerns is proposed in Section 4. An experimental evaluation of the proposed algorithm is presented in Section 5, together with a comparison of our approach with similar existing approaches. Some conclusions of the paper and further research directions are given in Section 6.

2 Related Work

Aspect mining is a relatively new research domain. However, many aspect mining techniques have been proposed. Some use metrics (Fanin [13]), some use formal concept analysis (Dynamo [24], Identifier analysis [25], History Mining [3]), or execution relations (DynAMiT [2]). There are also a few approaches that use clone detection techniques [4, 22] or natural
language processing [18].

A graph based approach in Aspect Mining is introduced in [19]. The basic idea of this technique is to determine methods that are similar. The approach is to construct a graph between the methods of the software system, to determine the connex components of this graph, called clusters, and then to identify crosscutting concerns in the obtained clusters.

There are just a few aspect mining techniques proposed in the literature that use clustering in order to identify crosscutting concerns [7, 16, 20, 23].

Shepherd and Pollock [23] have proposed an aspect mining tool based on clustering. They use hierarchical clustering to find methods that have common substrings in their names. The obtained clusters are then manually analyzed to discover crosscutting concerns. He and Bai [7] have proposed another aspect mining technique based on dynamic analysis. They also obtain execution traces for each use case, but they apply clustering and association rules to discover aspect candidates. Moldovan and Serban [16] have also proposed an aspect mining technique based on clustering. A vector space model based clustering is used in order to group methods in clusters. A part of the obtained clusters are then manually analyzed to discover crosscutting concerns. This approach is improved in [20], by defining a \( k\)-means based clustering algorithm in aspect mining (\( k\)AM). An evolutionary approach in aspect mining is introduced in [21] and two genetic clustering algorithms used to identify crosscutting concerns are proposed.

3 Partitional Clustering. The \( k\)-medoids clustering algorithm

Unsupervised classification, or clustering, as it is more often referred as, is a data mining activity that aims to differentiate groups (classes or clusters) inside a given set of objects [6], being considered the most important unsupervised learning problem. The resulting subsets or groups, distinct and non-empty, are to be built so that the objects within each cluster are more closely related to one another than objects assigned to different clusters. Central to the clustering process is the notion of degree of similarity (or dissimilarity) between the objects.

Let \( \mathcal{O} = \{O_1, O_2, \ldots, O_n\} \) be the set of objects to be clustered. The measure used for discriminating objects can be any metric or semi-metric function \( d: \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R} \). The distance expresses the dissimilarity between objects.

A well-known class of clustering methods is the one of the partitioning methods, with representatives such as the \( k\)-means algorithm or the \( k\)-medoids algorithm. Essentially, given a set of \( n \) objects and a number \( k, k \leq n \), such a method divides the object set into \( k \) distinct and non-empty clusters. The partitioning process is iterative and heuristic; it stops when a “good” partitioning is achieved. Finding a “good” partitioning coincides with optimizing a criterion function defined either locally (on a subset of the objects) or globally (defined over all of the objects, as in \( k\)-means). These algorithms try to minimize certain criteria (a squared error function); the squared error criterion tends to work well with isolated and compact clusters [8].

In \( k\)-medoids or PAM (Partitioning around medoids) algorithm [9], each cluster is represented by one of the objects in the cluster. It finds representative objects, called medoids, in clusters. The algorithm starts with \( k \) initial representative objects for the clusters (medoids), then iteratively recalculates the clusters (each object is assigned to the closest cluster - medoid), and their medoids until convergence is achieved. At a given step, a medoid of a cluster is replaced with a non-medoid if it improves the total distance of the resulting clustering [9].

4 A New Partitional Clustering Algorithm for Crosscutting Concerns Identification (PACO)

In this section we introduce a new partitional clustering algorithm (PACO) (Partitional Clustering Algorithm for Crosscutting Concerns Identification) for identifying crosscutting concerns in existing software systems. In order to discover the crosscutting concerns from the system, we first analyze the software system to be mined. All classes, methods and relations between them are computed. Afterwards, PACO algorithm is used to identify a partition of a software system \( S \) in which the methods belonging to a crosscutting concern should be grouped together. The final step is to manually analyzed the obtained results.

4.1 PACO algorithm

Let us consider that the software system to be mined consists of a set of classes \( \mathcal{C} = \{c_1, c_2, \ldots, c_k\} \), each class containing one ore more methods. In our clustering approach, the objects to be clustered are the methods from the software system \( S \), i.e., \( \mathcal{M} = \{m_1, m_2, \ldots, m_n\} \). Our focus is to group the methods such that the methods that belong to the same crosscutting concern to be placed in the same cluster.

In order to apply a clustering algorithm, we will consider the dissimilarity degree between any two methods from the software system \( S \). Consequently,
we will consider the distance \( d(m_i, m_j) \) between two methods \( m_i \) and \( m_j \) as expressed in Equation (1).

\[
d(m_i, m_j) = \left\{ \begin{array}{ll} 1 - \frac{|\text{in}(m_i) \cap \text{in}(m_j)|}{|\text{in}(m_i) \cup \text{in}(m_j)|} & \text{if } \text{in}(m_i) \cap \text{in}(m_j) \neq \emptyset \\
\infty & \text{otherwise} \end{array} \right.
\]

where, for a given method \( m \in \mathcal{M} \), \( \text{in}(m) \) defines a set of methods and classes that invoke \( m \), as expressed in Equation (2).

\[
\text{in}(m) = \{ m' \in \mathcal{M} \cup \mathcal{C} | \text{m'} \text{ invoke } m \}. \tag{2}
\]

In our view, the distance between two methods as defined in (1) expresses the following idea: if two methods are invoked by common methods or classes, they should belong to the same cluster. This means that scattered methods would be placed in the same cluster, and the methods that do not represent aspects would not be placed together with methods from aspects as the latest are invoked from multiple places. Consequently, methods that belong to the same crosscutting concern are close (considering distance \( d \)) to each other.

Many programming languages allow the definition of inner classes, classes that are defined inside of another class. This grouping of classes frequently appears in real life software projects, and our distance consider this situation, also. The set \( \text{in}(m) \) will contain not only the class \( c \) that invoke the method, but also the class that contains class \( c \).

Based on the definition of distance \( d \) (Equation (1)) it can be easily proved that \( d \) is a semi-metric function, so a \( k \)-medoids based approach can be applied.

In order to avoid the two main disadvantages of the traditional \( k \)-medoids algorithm, \( \text{PACO} \) algorithm uses a heuristic for choosing the number of medoids (clusters) and the initial medoids. This heuristic is particular to aspect mining and it will provide a good enough choice of the initial medoids.

After selecting the initial medoids, \( \text{PACO} \) behaves like the classical \( k \)-medoids algorithm.

The main idea of \( \text{PACO} \)'s heuristic for choosing the initial medoids and the number \( p \) of clusters (medoids) is the following:

(i) The initial number \( p \) of clusters is \( n \) (the number of methods from the software system).

(ii) The method chosen as the first medoid is the most “distant” method from the set of all methods (the method that maximizes the sum of distances from all other methods).

(iii) In order to choose the next medoid we reason as follows. For each remaining method (that was not chosen as medoid), we compute the minimum distance (\( d_{\text{min}} \)) from the method and the already chosen medoids. The next medoid is chosen as the method \( m \) that maximizes \( d_{\text{min}} \) and this distance is greater than a positive given threshold (\( d_{\text{distMin}} \)). If such a method does not exist, it means that \( m \) is very close to all the medoids and should not be chosen as a new medoid. From the aspect mining point of view this means that \( m \) should belong to the same (crosscutting) concern with an existing medoid. In this case, the number \( p \) of medoids will be decreased.

(iv) The step (iii) will be repeatedly performed, until \( p \) medoids will be reached.

We have to notice that step (iii) described above assures, from the aspect mining point of view, that near methods (with respect to the given threshold \( d_{\text{distMin}} \)) will be merged in a single (crosscutting) concern (cluster), instead of being distributed in different (crosscutting) concerns.

We mention that at steps (ii) and (iii) the choice could be a non-deterministic one. In the current version of \( \text{PACO} \) algorithm, if such a non-deterministic case exists, the first selection is made. Future improvements of \( \text{PACO} \) algorithm will deal with these kind of situations.

The main idea of the \( \text{PACO} \) algorithm that we apply in order to group methods from a software system is the following:

(i) The initial number \( p \) of clusters and the initial medoids are determined by the heuristic described above.

(ii) The clusters are recalculated, i.e., each object is assigned to the closest medoid.

(iii) Recalculate the medoid \( i \) of each cluster \( k \) based on the following idea: if \( h \) is an object from \( k \) such that \( \sum_{j \in k} (d(j, h) - d(j, i)) \) is negative, then \( h \) becomes the new medoid of cluster \( k \).

(iv) Steps (ii)-(iii) are repeatedly performed until there is no change in the partition \( K \).

We mention that \( \text{PACO} \) algorithm provides a partition of a software system \( S \), partition that ideally would contain separate clusters for each crosscutting concern.

Regarding to \( \text{PACO} \) algorithm, we have to notice the following:

- If, at a given moment, a cluster becomes empty, this means that the number of clusters will be decreased.
• Because the initial medoids are selected based on the heuristic described above, the dependence of the algorithm on the initial medoids is eliminated.

• We have chosen the value 1 for the threshold $\text{distMin}$, because distances greater than 1 are obtained only for unrelated entities (Equation (1)). Our intuition for choosing the value for the threshold $\text{distMin}$ was experimentally confirmed. In the future we plan to find the most appropriate value for the threshold $\text{distMin}$ using supervised learning techniques [14] and to give a rigorous proof for our selection.

5 Experimental Evaluation

In this section we want to evaluate how well did PACO algorithm succeed in grouping the elements from crosscutting concerns in clusters. For that we have considered the open source case study JHotDraw, version 5.4b1 [5]. It is a Java GUI framework for technical and structured graphics, developed by Erich Gamma and Thomas Eaggenschwiler, as a design exercise for using design patterns. It consists of 396 classes and 3359 methods.

The set of crosscutting concerns used for the evaluation is: Adapter, Command, Composite, Consistent behavior, Contract enforcement, Decorator, Exception handling, Observer, Persistence, and Undo. The set of crosscutting concerns and their implementing methods was constructed using the results reported by Marin et al. and publicly available at [12].

In order to evaluate the results we use a quality measure, called $\text{DISP}$, that we have previously introduced in [15]. This measure defines the dispersion degree of crosscutting concerns in clusters, considering, for each crosscutting concern, the number of clusters that contain elements belonging to the concern.

We give below the formal definition of $\text{DISP}$ measure. In the following $\text{CCC}$ denotes the set of crosscutting concerns existing in a software system, $\mathcal{K}$ denotes a partition of the set $\mathcal{M}$ of methods from the software system to be mined. The partition $\mathcal{K}$ can be obtained using a clustering algorithm (PACO in this paper).

**Definition 1** [15] $\text{DISPersion of crosscutting concerns - DISP}$

The dispersion of the set $\text{CCC}$ of crosscutting concerns in the partition $\mathcal{K}$, denoted by $\text{DISP}(\text{CCC}, \mathcal{K})$, is defined as

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

(3)

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

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

(4)

where

$$D_C = \{k | k \in \mathcal{K} \land k \cap C \neq \emptyset\}.$$  

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

The values of the $\text{DISP}$ measure are in the interval $[0, 1]$. The proof can be found in [15]. In order to obtain more cohesive partitions, $\text{DISP}$ measure has to be maximized.

After applying PACO algorithm for JHotDraw case study we have obtained a partition $\mathcal{K}$ with $\text{DISP}(\text{CCC}, \mathcal{K}) = 0.4444$.

We have compared the results obtained by PACO algorithm with the results obtained by $\text{kAM}$ algorithm proposed in [20]. $\text{kAM}$ algorithm is based on the idea of $k$-means clustering and uses a heuristic for choosing the initial centroids and the initial number of clusters. The similarity between two methods is computed using a vector space model based approach. The value of the $\text{DISP}$ measure obtained by $\text{kAM}$ for the same case study is 0.4005.

Comparatively, considering the $\text{DISP}$ measure, PACO algorithm has obtained better results than $\text{kAM}$ algorithm. This means that in the partition obtained by PACO algorithm the methods from the crosscutting concerns were better grouped than in the partition obtained by $\text{kAM}$ algorithm. However, the elements of crosscutting concerns are spread in two or more clusters of a partition for both algorithms, as the values of the $\text{DISP}$ measure are less than 0.5 for both PACO and $\text{kAM}$ algorithms.

We did not provide a comparison of the considered approach with the two other existing clustering based aspect mining approaches for the following reasons:

• Shepherd and Pollock have proposed in [23] an aspect mining tool based on clustering that does not automatically identify the crosscutting concerns. The user of the tool has to manually analyze the obtained clusters in order to discover crosscutting concerns.

• The technique proposed by He and Bai [7] cannot be reproduced, as they do not report neither the clustering algorithm used, nor the distance metric between the objects to be clustered. Also, the results obtained for the case study used by the authors for evaluation are not available.
6 Conclusions and Future Work

We have presented in this paper a new partitional clustering algorithm (PACO) that can be used for identifying crosscutting concerns in existing software systems. The proposed algorithm uses an heuristic for choosing the number of clusters and the initial medoids, reducing this way the disadvantages of the traditional k-medoids method.

In order to evaluate the obtained results, we have considered JHotDraw case study.

Further work can be done in the following directions:

- To apply other clustering techniques and to improve PACO algorithm.
- To use other approaches for clustering that were proposed in the literature (such as search based clustering [11,1] or genetic clustering).
- To improve the distance semi-metric used for discriminating the methods in the clustering process in order to consider the tangling symptom, also.

Acknowledgements: This work was supported by the research project ID 2268/2009, sponsored by the Romanian National University Research Council (CNCSIS).

References:


