Crosscutting Concern Identification Using Graph Theory

Richa Singh  
Banaras Hindu University

Tapan Kant*  
Banaras Hindu University

Manjari Gupta  
Banaras Hindu University

Abstract
The process of determining crosscutting candidates through aspect mining and refactoring those candidates in existing software system is, for a good AOP [1]. In this paper we proposed a graph based technique for identifying the crosscutting candidates. We are providing in initial assessment from the existing [2] how a vector space model and identification of crosscutting concerns steps proposed our algorithm based on crosscutting concerns graph algorithm. This algorithm is used to identify the concerns in the system on the basis of distance between methods in the graph.

Index Terms—Aspect Mining, Aspect Oriented Programming, Crosscutting Concern, Vector-Space-Model, Object-Oriented-Programming.

[1] INTRODUCTION
Aspect oriented programming is a new prototype that is initiated to design and execute crosscutting concerns [1]. AOP mainly tackle the modularity limitations of Object oriented programming (OOP). It attempts to effectively sum up the crosscutting concerns to form a new type of module known as an aspect [19]. An Aspect is a modular unit of crosscutting implementation. It is defined very much like a class, and can have methods, fields, constructors, initializers, named pointcuts, and advices [1]. At compile time, the aspect is weaved to generate the final system, using a special tool called weaver. Crosscutting concerns are characteristics of software system whose implementations are spread over the source code. Crosscutting concerns in non AO have two symptoms. Code Scattering: the code for one concern is spread across the system. Code tangling: in order to implement one concern the programmer needs to address or crosscut another concerns as well. Aspect mining is a process that tries to identify crosscutting concerns in existing software systems. The goal is to refactor the existing systems to use aspect oriented programming to make them easier to maintain and to evolve [3].

[1.1] Aspect Mining
Aspect mining is a relatively new research direction that tries to identify crosscutting concerns in already developed software systems without using AOP [3]. 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. Aspect Mining is a reverse software engineering exploration technique that is concerned with the development of concepts, principles, methods and tools supporting the identification and extraction of re-factorable aspect candidates in legacy software systems [4]. To achieve this level of modularization in legacy and traditional object-oriented applications, candidates for refactoring first need to be identified. This step is known as aspect mining [5]. In other words, aspect mining refers to the discovery of potential crosscutting concerns that can be modularized into aspects. Through Aspect mining technique we can identify crosscutting concerns during software development lifecycle or in already developed software system. The goal of these techniques is to provide an overview of the source-code entities that play a role in a particular crosscutting concern. This not only improves the understandability of the concern in particular and of the software in general, but also provides a first step in the migration towards applying aspect-oriented software development techniques [5].
Our approach uses clustering based aspect mining techniques. There are just a few aspect mining techniques which uses clustering to identify crosscutting concerns [2, 3, 4, 11, 13, 15].

In our approach a connected weighted graph is constructed for a given source code where vertices are corresponding to methods and we assume among all methods weights of edges are corresponding to the Euclidian distance between two methods. By applying given approach described in this paper different subcomponents of this connected graph are identificies. These different subcomponents are corresponding to different crosscutting concerns in the source code. For this we will consider a vector space model, identification of crosscutting concerns steps and threshold value described in [2].

The paper is structured as follows, section 2 describes related work. Section 3 and section 4 describes a vector space model and identification of crosscutting concerns steps [2] which are used in our approach. Section 5, describes our approach for identification of crosscutting concerns and section 6 describes the system graph algorithm. In the last section 7 we conclude and present future work.

[2] RELATED WORK

Several approaches have been proposed for aspect mining till now. The first approaches in aspect mining were the query based search techniques. The developer had to search so called and associated tool showed all the places where the seed was found [6]. There are many tools have been developed for this as Aspect mining tool [7], Aspect browser [8], AMTEX [9]. Feature Exploration and analysis tool(FEAT)[10].

Later on researchers have focused on developing such aspect mining techniques that do not require initial seed from the user [6]. Some of them uses fan in analysis techniques [3, 11], some of them uses Formal concept analysis [5], some of them use clustering ([3], [12], [13]) some of them uses clone detection techniques [14]. Those techniques try to identify the crosscutting concerns with some kind of system representation and are called automated aspect mining techniques [6]. These are all static approaches that analyze the source code for crosscutting concerns. In [5] a dynamic approach was proposed to analyze the execution of source code to identify crosscutting concerns.

In [18], proposed a clustering approach in aspect mining based on k-means and hierarchical agglomerative clustering techniques and in [17] some quality measures for evaluating the results of clustering based aspect mining techniques.

[3] VECTOR SPACE MODEL

Several approaches have been used vector space model [2] for their own purpose. We are also used the vector space model described in the below: In our approach the vertices to be considered are the methods from the software system M = {m1, m2, ...mn}. The methods belong to the application classes or are called from the application classes. Based on the vector space model, [2] considered each method as an l-dimensional vector as M = {m1, m2, ... mn} two vector-space models have considered that illustrate code scattering symptoms in software system [2]:

1-The vector associated with the method m is \{FIV, CC\}, where FIV is the fan-in value [11] of m (the number of methods that call m) and CC is the number of calling classes.

This model has been denoted by M1.

2- The vector associated with the method m is \{FIV, B1, B2, ... Bn\}, where FIV is the fan-in value of m and Bi(1 ≤ i ≤ l-1) is 1, if the method m is called from a method belonging to the application class Ci, and 0, otherwise. We denote this model by M2.

In our approach we will consider Euclidian distance as a weight of the edges between two vertices (methods) in the connected graph. In this context the Euclidian distance between two methods m1 and m2 defined as:-

\[ d_{e}(m_{1}, m_{2}) = \sqrt{\sum_{k=1}^{l}(r_{i,k} - r_{j,k})^2} \]

In our approach we have chosen the value 1 for the threshold. The reason for this choice is the following [2]: if the distance between two methods m1 and m2 is less or equal to 1, considered that they are similar enough to be placed in the same (crosscutting) concern. It mentioned that, from the aspect mining point of view, using Euclidian distance as metric and the vector space model proposed above, the value 1 for threshold makes the difference between a crosscutting concern and a non-crosscutting one.

[4] IDENTIFICATION OF CROSSCUTTING CONCERNS STEPS

The following steps [2, 3, 15 & 16] are defined for identification of crosscutting concerns.
Step 1: Computation of set of methods in the chosen source code as well as computation of the features set values, for each method in the set.

Step 2: Filtering techniques owned by certain data structures categories like ArrayList, vectors are removed. Also, the techniques owned by some built-in categories like String, String Builder and String Buffer are removed.

Step 3: By using System graph algorithms described in section 5, the remaining set of methods is arranged into clusters. The clusters are separated by the average distance from the point 0m in descending order, where 0m is referred to m dimensional vector with individual part 0.

Step 4: The clusters acquired are evaluated to detect which methods belong to crosscutting concerns. The cluster that contain procedures owned by crosscutting concerns [11].

The Algorithm that would be used in step 3 we are purposed in the next section.

[5] OUR APPROACH

Let X = {m₁, m₂, ..., mₙ} be a software system, where 1 ≤ n is a method of the system. We consider a crosscutting concerns as a set C = {c₁, c₂, ..., cₘ} with C ⊂ m, of methods that implement this concerns. The number of methods in crosscutting concerns c is C_c = |C|, Let CCC = {C₁, C₂, ..., Cₚ} be the set of all crosscutting concerns that exist within X. The number of crosscutting concerns in the system X is therefore represented by q=|CCC|.

In this section we describe the concept of system Crosscutting Concern using Graph and the following definitions are needed to describe our approach.

Definition 1:- Let X = {m₁, m₂, ..., mₙ} be software methods and dₑ the Euclidian distance metric between edges (methods). We can define System graph Sₓ as: Sₓ=(V,E) ; Where the set of vertices V={m₁, m₂, ..., mₙ} of the software system X. And the set of edges E={(mᵢ,mⱼ) | mᵢ,mⱼ ∈ V, mᵢ≠mⱼ} & all the edges are labeled with the Euclidian distance as:

\[ W(mᵢ,mⱼ) = dₑ(mᵢ,mⱼ) = \sum_{k=1}^{l}(xᵢₖ - xⱼₖ)^2 \]

Definition 2:- Let X={m₁, m₂, ..., mₙ} be a software system and dₑ (Euclidian distance) the metric between methods in a multidimensional and Sₓ the corresponding system graph (Definition 1). We define the problem of finding those components whose edges Euclidian distance is less than one i.e. dₑ(mᵢ,mⱼ)<1, in the software system X. Algorithm for finding those components of crosscutting concerns from system graph Sₓ is described in the next section.

[6] SYSTEM GRAPH ALGORITHM

In this part we describe Sₓ algorithm for determining the dₑ(mᵢ,mⱼ)<1, of software system X. This algorithm will be used in grouping step in the section 4 [2] for identification of crosscutting concerns. These are the following steps in Sₓ algorithm:-

1-Create the concerns graph as shown in definition 1.
2-Determine the edges whose distance between vertices (methods) is less than 1. These edges are concerns in the software system X.

[7] CONCLUSIONS AND FUTURE WORKS

We have presented in this paper a new graph based approach in aspect mining. For this we have used system concerns algorithm, in the grouping step of identification of crosscutting concerns. We will find the connected graph whose Euclidian distance between vertices (methods) is less than one and obtained vertices are analyzed in order to identify crosscutting concerns from the system.

Future work can be done in the following points:

- We can apply this approach in case studies, like JHOT Draw, Pet store and JEdit etc.
- We can compare the results with other algorithm of aspect mining techniques.
- We can improve vector space model to obtained better result using this algorithm.
- We can take another threshold value to improve the result.

REFERENCES


