Advances in Engineering Software 41 (2010) 1160–1178

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Software refactoring at the function level using new Adaptive K-Nearest Neighbor algorithm

Abdulaziz Alkhalid [a], Mohammad Alshayeb [b,*], Sabri Mahmoud [b]

[a] 4700 King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
[b] Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

ARTICLE INFO

Article history:
Received 2 April 2010
Received in revised form 27 June 2010
Accepted 5 August 2010

Keywords:
Software refactoring
Clustering
Cohesion
Code restructuring
Function level cohesion
Software quality

ABSTRACT

Improving the quality of software is a vital target of software engineering. Constantly evolving requirements force software developers to enhance, modify, or adapt software. Thus, increasing internal complexity, maintenance effort, and ultimately cost. In trying to balance between the needs to change software, maintain high quality, and keep the maintenance effort and cost low, refactoring comes up as a solution. Refactoring aims to improve a number of quality factors, among which is understandability. Enhancing understandability of ill-structured software decreases the maintenance effort and cost. To improve understandability, designers need to maximize cohesion and minimize coupling, which becomes more difficult to achieve as software evolves and internal complexity increases. In this paper, we propose a new Adaptive K-Nearest Neighbor (A-KNN) algorithm to perform clustering with different attribute weights. The technique is used to assist software developers in refactoring at the function/method level. This is achieved by identifying ill-structured software entities and providing suggestions to improve cohesion of such entities. We also compare the proposed technique with three function-level clustering techniques Single Linkage algorithm (SLINK), Complete Linkage algorithm (CLINK) and Weighted Pair-Group Method using Arithmetic averages (WPGMA). A-KNN showed competitive performance with the other three algorithms and required less computational complexity.

1. Introduction

Changing requirements lead real-world software to constantly evolve over time. As the software is enhanced, modified and adapted, the code becomes more and more complex. Thus, the quality of the software decreases. The major part of the total software development cost is devoted to software maintenance [1–3]. This creates a need for techniques that reduce software complexity, and hence, improve quality and reduce maintenance cost. One such technique is referred to as restructuring [4,5] or, refactoring [6,7] in Object Oriented (OO) software development. In a disciplined way, refactoring restructures the code into a simpler form to improve its internal structure without changing its external functionality or behavior.

Refactoring aims to improve several factors of software quality such as understandability. It also makes developers program faster and helps in finding bugs [7]. Since refactoring changes the internal structure of the code, the internal quality attributes such as coupling and cohesion will change [8]. Software designers try to maximize cohesion while minimize coupling [9]. The task of balancing cohesion and coupling becomes challenging as software evolves over time causing higher levels of internal complexity [9]. To aid software designers in this increasingly difficult mission, some automated techniques emerged as possible solutions. Pattern recognition based techniques can give suggestions to enhance the cohesiveness of software components during software refactoring activities.

Clustering is a statistical pattern recognition method used for organizing data. A pattern is the opposite of chaos. Recognition is the process of identifying a pattern as a member of a category. It is the act of taking in raw data and taking an action based on the "category" of the pattern [10]. Different models of classification can be used in pattern recognition systems such as Template Matching, Statistical (geometric), Syntactic (structural), Artificial Neural Network (biologically motivated), and Hybrid approach. These models require different types of features or sets of features and may have different application areas. The statistical pattern recognition approach focuses on the statistical properties of the patterns, which are generally expressed in probability densities. Thus, the patterns are usually represented in a feature space and the goal is to partition the feature space [10].

* Corresponding author.
E-mail addresses: abdulaziz.alkhalid@kaust.edu.sa (A. Alkhalid), alshayeb@kfupm.edu.sa (M. Alshayeb), smasaad@kfupm.edu.sa (S. Mahmoud).

0965-9978/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2010.08.002

In this paper, we present an approach to software refactoring at the function/method level using clustering algorithms. The objective is to provide automated support to identify ill-structured or low-cohesive functions and present refactoring suggestions as heuristic advice to software designers to guide their refactoring activities.

We introduce a new Adaptive K-Nearest Neighbor (A-KNN) algorithm to perform clustering with different attribute weights. We also compare its performance to other three other hierarchical agglomerative clustering algorithms: Single Linkage algorithm (SLINK), Complete Linkage algorithm (CLINK) and Weighted Pair-Group Method using Arithmetic averages (WPGMA).

Table 1
Attributed matching combinations and their indications.

Combination	Indication
0–0 match	The data or control attribute are not present in both entities
1–1 match	The same control attribute is present in both entities.
2–2 match	The same data attribute is present in both entities and neither of them is a control entity
1–0 or 0–1 match (mismatch)	A control attribute is present in one entity and is not present in the other
2–0 or 0–2 match (mismatch)	A data attribute is present in one entity and is not present in the other
2–1 or 1–2 match	A data attribute is present in both entities. However, it is a control variable in one entity, and it is a non-control variable in the other (like the variable which contains the maximum number of iterations in a loop definition)

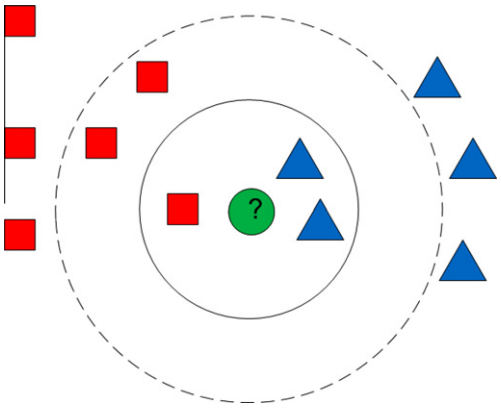

Fig. 1. Example of KNN classification.

2. Literature review

Researchers suggested using clustering techniques to restructure software systems. Wiggerts [11] provided an overview of clustering analysis and of system re-modularization. Tzerpos and Holt [12] surveyed clustering approaches and concluded that clustering techniques used in other disciplines can be used in the software context. Lakhotia [13] presented a unified framework for expressing methods of classifying subsystems of a software system. The framework helps in comparing various subsystem classification techniques.

Other researchers proposed and used different clustering techniques to restructure modules. Kim and Kwon [14] present methods that can be readily automated to restructure ill-structured modules at the maintenance phase. They used the module strength as a criterion to decide how to restructure the module. Kang and Beiman [15,16] introduced a quantitative framework for software restructuring. They used measures of design-level cohesion and coupling as the criteria for comparing alternative design structures. Lakhotia and Deprez [17,18] proposed a transformation called "tuck" to restructure programs by breaking large functions into small functions. The method complements those reported in [14–16]. Lung et al. [19] applied clustering techniques to functional restructuring and demonstrated how to restructure a low-cohesive function into high-cohesive functions. They treated executable program statements as basic entities and variables as attributes.

Komondoor and Horwitz [20] described an algorithm for extracting "difficult" sets of statements and compared the algorithm previously reported automatic approaches and to human extraction. Harman and Hierons [21] reviewed three forms of program mslicing: static, dynamic and conditioned and two syntactic paradigms: syntax–preserving and amorphous. Maruyama [22] proposed a technique that automatically refactors of object-ori-

```
Algorithm: A-KNN for K=3
Input: Entity-Attribute Matrix (n * m); n: number of entities, m: number of attributes
Output: Hierarchy of clusters.
Steps:
   1. Assign each entity to a single cluster and label each cluster. (L (Ci)=unique
      label, i belongs to {1,..,n}; where n is the number of entities)
   2. Calculate the Similarity Matrix (n * n) (Calculate the similarity between each
      cluster and all other clusters using the formula (1) and fill the n * n matrix).
   3. While the number of clusters is more than one Do
      a. Find the most similar three pairs of clusters {Ca, Cb}, {Cd, Ce}, {Cf, Cg},
         where: Coeff (Ca, Cb) > Coeff (Cd, Ce) > Coeff (Cf, Cg)
      b. If L (Ca) = L (Cd) = L (Cf) ( the same cluster) Then
         i. If L (Ce)= L (Cg) Then
            1. L (Ca)= L (Ce) (merge clusters of Ca and Ce in one cluster)
        ii. Else
            1. L (Ca)= L (Cb) (merge clusters of Ca and the Cb in one cluster (normal
               clustering))
      c. Else
         i. L (Ca)= L (Cb) (merge clusters of Ca and Cb in one cluster (normal
            clustering))
   4. End While
   5. Return 'the hierarchy of clusters'
```

Fig. 2. A-KNN algorithm for $K = 3$.