SURVEY ON SOFTWARE REMODULARIZATION TECHNIQUES

Rajalakshmi.M[1], Bright Gee Varghese.R[2]

[1] *Post-Graduate Student, Department of Computer Science and Engineering, Karunya University, India,itisrebecca@gmail.com*

[2] *Assistant professor , Department of Computer Science and Engineering, Karunya University, India,brighfsona@yahoo.co.in*

Abstract

Re-modularization of software systems is a key technique used in testing and maintenance phase of the software development lifecycle. This process helps the developer in reviewing the modules and the interaction between those modules of the developed software systems. Re-modularization process also allows the developer to make the software system more efficient by adding additional features according to their future requirement and enhancement. Based on the review results, the developed software system can be re-modularized, thus finally making the software system to be efficient and less fault-prone. This paper deals with the survey on various methods that are used for re-modularization of software systems. The methods include hierarchical clustering method, automated clustering methods and clustering using Genetic Algorithm (GA).The usage of hierarchical clustering helps in understanding the software domain, software system and the measures used for clustering the software system. Automatic clustering approaches are used in order to group together highly cohesive components, in a module of the software system where the cohesiveness is measured in terms of intra-module links and reduce the coupling between modules which is measured in terms of the inter-module dependencies of the software system. Genetic Algorithm belongs to large class of Evolutionary Algorithm, which generates the solutions to optimization problems using techniques inspired by natural evolution such as inheritance mutation, crossover and selection. In general, Genetic Algorithm requires the genetic representation of the solution domain and the fitness function to evaluate the solution domain. In this survey the working principle of various re-modularization methods are discussed and the efficient re-modularization approach is selected based on its features and advantages.

Index Terms: *Software re-modularization, Hierarchical Clustering and Genetic Algorithm.*

----------***----------

1. INTRODUCTION

Software systems in general consist of modules and methods that interact with each other in order to accomplish the purpose for which those systems are actually developed. The unchanging fact is that these developed software systems are exposed to modifications or changes. This may be done in the view of detecting and correcting errors or in need of improving the efficiency of software systems by introducing additional features based on their future requirements. This is termed to be the re-modularization process of the software systems. [1] (Fowler .M et al. 1999) says the modifications made in the developed software may however reduces the cohesiveness of the modules and increases the coupling between various modules and thus making the resultant software system to be harder to maintain and possibly be more fault-prone.

To overcome this situation (i.e.) to improve cohesion with the module in the software system and to reduce coupling between various modules within the software systems, various re-modularization techniques are used. Hierarchical clustering based techniques [2], automated clustering approaches [3,4] and clustering using Genetic Algorithm(GA) [5,6] are some ways in which re-modularization process of software systems can be done. The resultant package or modules of the software system will possess high cohesiveness and low coupling characteristics.

Apart from these techniques we also discuss some other techniques like re-modularization based on structural and semantic metrics [7], clustering based on frequent common changes [8] and supervised software re-modularization process[9].

2. HIERARCHICAL CLUSTERING METHODS

Hierarchical clustering is a statistical tool that is most commonly used for discovering the relationships in statistical data. Distances, Similarities, Correlation are some of the parameters based on which clustering is done. The strategies of hierarchical clustering mainly falls into two types:

Agglomerative: In this type, a single cluster is the initial point of each observation and while moving up the hierarchical structure, pairs of clusters are merged. This is known to be the "bottom up" approach.
Divisive: In this type all observations are started in one common cluster and then it is splitted recursively down the hierarchical structure. This is known to be the "top down" approach.

In [2] (Maqbool. O et al. 2007) proposed a method for hierarchical clustering and the importance of software architecture in software system's organization. The approach used in this paper involves the following steps
Identification of entities: For traditional legacy systems, functions are used as entities, since they are the basic components of the system. For very large systems, source files are recommended to be used as entities.
Selection of features: Functions called by an entity, global variables referred by an entity and user defined types accessed by an entity forms the formal features of the software system.
Selection of similarity measures: The Jaccard, Ellenberg and information loss measures were used as similarity measures.
Selection of a clustering algorithm: Clustering is done based on the agglomerative approach.
Selection of evaluation methods and measures: The following methods and measures are used in evaluating the clustering results:
Internal assessment: Various clustering algorithms are used to obtain the decomposition of the system automatically and the resultant decompositions are evaluated and cluster stability is also evaluated.
External assessment: The quality of results obtained by various clustering algorithm is evaluated by comparing it with decompositions prepared manually by human experts.
Relative assessment: More than one expert was asked to prepare decompositions for test system and the results were compared. This helps in determining the functions whose placement in a subsystem is difficult to determine.

The major advantage in this paper is the comparison of various clustering algorithms helps in deeper understanding of the software domain, software systems and measures employed for clustering.

3. AUTOMATED CLUSTERING METHODS

In order to increase the cohesiveness and reduce the coupling characteristics of the modules within the package of the software system, several automatic clustering approaches are employed.

In [3] (Maqbool O, Babri H.A et al.2006) says clusters obtained as a result of the clustering algorithms may not more helpful without labeling those clusters. The assignment of labels can be done automatically. This paper during the process of clustering the keyword within the identifiers is ranked. The ranking is done using two approaches: frequency and inverse frequency. The main objective of this paper is to define a labeling scheme based on function identifiers of an entity. This paper initially presents a cluster labeling scheme developed based on identifiers of an entity. Then it compares the clustering process of the complete and weighted combined algorithms using cluster labels. Finally, it compares the frequency and inverse frequency ranking schemes for cluster labeling. Instead of carrying out external assessment, a comparison of the complete and weighted combined algorithms is done by evaluating their clustering processes during hierarchical clustering. The Frequency based approach of cluster labeling includes the following steps. Initially the selection of entities and label section is done. Then the similarity measures of the clusters are selected. Finally the selection of clustering algorithm is done. Then the cluster labeling process which includes keyword selection and keyword ranking is employed. The major advantage of this paper is that the labels obtained from this technique are meaningful and helpful in gaining the knowledge of why the software system is actually developed and the purposes of the sub-systems in the software system.

In [4] (Hani Abdeen et al. 2009) says, for a software system to support the replacement of its parts without affecting the whole system, it should be a well modularized system. But the fact is that when the software system evolves over time with modifications, the modularization gradually drifts and the quality of the software system will also be lost to some extent. As a consequence some classes may not be placed in suitable packages. Optimizing the package structure and connectivity is generally required in order to improve the quality of the software system modularization. The objective of this paper is to automatically optimize the decomposition of software system into packages so that the resulting organization of classes/packages, mainly, reduces connectivity and cyclic-connectivity between packages**.** The implementation process of this paper is done using the concept of simulated annealing which is a neighbor search based technique. An optimization procedure is used to optimize the package connectivity that starts with a given modularization and gradually modifies it, using slight deviation of the actual system. At each step, the resulting modularization is evaluated to be possibly selected as an alternative modularization. Then the Modularization Quality

(MQ) is evaluated. In addition to the fitness function, our approach allows maintainers to define distinct constraints that should complete the evaluation process and guarantee maintainers' requirements. This paper enables maintainers to optimize object oriented (OO) package structure of source code. An optimization algorithm and as set of measures are used in optimization process automatically evaluate the quality of a modularization. These are some of the advantages of this paper.

4. CLUSTERING USING GENETIC ALGORITHM

Genetic Algorithm is simply the algorithm used to simulate evolution. It takes candidate solutions, selects some of the best solutions using user-defined evaluation functions, applies user-defined transformations such as mutation and crossover and finally makes a new candidate solution.

[5] (D. Doval et al.1999) proposed a paper on Automatic Clustering of Software Systems using a Genetic Algorithm. In general, large software systems tend to have a rich and complex structure. One way of making them more accessible is to partition them, separating their modules into clusters. The main goal is to find a good Module dependency graph (MDG) partitions as an optimization problem using Genetic Algorithm (GA). To find the high-level partition of the MDG Genetic Algorithm (GA) is used. GA operates on a set (population) of strings (*individuals*), where each string is an encoding of the problem's input data. Each string's *fitness* (quality value) is calculated using an objective function. It provides relatively independent clusters that contain highly inter-dependent modules. The Input for this approach is System's MDG and the Output is partition of that MDG. This is can be done iteratively which involves the following steps. Initially GA generates the initial population, creating random strings of fixed size. Then GA creates a new population by applying the selection and reproduction operator to select pairs of strings. The number of pairs will be the population size divided by two, so the population size will remain constant between generations. Later GA applies the crossover operator to the pairs of strings of the new population and it also applies the mutation operator to each string in the new population. Finally, GA replaces the old population with the newly created population and if the number of iterations is less than the maximum, go to step 2. Else, stop the process and display the best answer found. Genetic Algorithm (GA) is used to overcome some drawbacks in traditional search methods like lack of maintaining the history etc.

[6] (Paolo Tonella et al. 2010) says, in software development lifecycle the requirements prioritization occupies a vital role. The process of requirements prioritization can be viewed as the process of finding an order relation on the set of requirements under analysis. The order in which requirements are implemented in a system affects the value delivered to the final users in the successive releases of the system. The main objective of this paper is to minimize the user decision-making effort and to increase the accuracy of the final requirements ranking. For implementing this, pairwise comparisons are used to resolve the ties in prioritizing the requirements. An interactive genetic algorithm is used to achieve minimization in disagreement between a total order of prioritized requirements and the various constraints that are either encoded with the requirements or that are expressed iteratively by the user during the prioritization process. It also takes the advantage of interactive input from the user whenever the fitness function cannot be computed precisely based on the information available. This approach improves non interactive optimization, ignoring the elicited preferences, and it can handle a number of requirements. These are some of the advantages of this technique.

[7] (Bavota. G, et.al. 2010) proposed a method to automatically re-modularize the packages using structural and semantic measures to decompose a package into smaller to make them more cohesive ones. The proposed approach takes as input a package identified in the software as a candidate for re-modularization. Then, a measure reflecting a relationship (structural and semantic) between pairs of classes from the package is computed. The measured values between classes are stored in a $n \times n$ matrix, called class-by-class matrix, where n is the number of classes in the package under analysis. An entry $m_{i,j}$ in the class-by-class matrix represents the likelihood that class c_i and class c_j should be in the same package. Using the information in the class-by-class matrix the approach extracts chains of strongly related classes. The classes of the original package are distributed in different packages according to the extracted chains. If the number of extracted chains is one, no re-modularization is required. Otherwise, based on the extracted class chains the approach suggests new packages with higher cohesion than the original package. The structure of individual classes in the package is not changed. The advantage of the proposed approach analyzes the relationships between classes in a package identifying chains of strongly related classes. The identified chains are used to define new packages with higher cohesion than the original package.

[8] (Dirk Beyer et al. 2005) says, abstract descriptions of a large software system enable software engineers to modify or extend the system without understanding every part of it in detail. When design documents with such descriptions are unavailable or out of date, high-level descriptions can be recovered from the source code and other low-level information through reverse engineering. As a part of this process, software clustering divides software artifacts into subsystems, such that the subsystems are as independent as possible with respect to comprehension, change, reuse, or other criteria. The software clustering method used in this paper includes a co-change graph and a novel clustering method for co-change graph. The vertices of the co-change graph are software artifacts and change transactions, and its edges connect the change transactions with their participating artifacts. The co-change graph can be easily extracted from version repositories. The novel clustering method used in this paper is energy-based graph clustering which results in layouts rather than producing partitions. The advantages of the process used are, it is simple and its direct correspondence to the modeled version repository ensures that the clustering results have a clear interpretation in terms of the repository, and biases through arbitrary choices (e.g., for weight functions or values of free parameters) are minimized.

[9] (Mathew Hall et al. 2012) proposed a paper on, 'Supervised software modularization'. The objective of this paper is to develop an approach that should obey good design principles (e.g. minimal coupling and maximal cohesion) and should also make sense to the developer or maintainer. This paper is implemented using SUMO algorithm. The SUMO (Supervised Re-modularization) algorithm provides a process within which it becomes possible to iteratively feed domain knowledge into the re-modularization process. Although it does not rely upon clustering techniques, it can benefit from them in the sense that outputs from existing tools can be used as starting points for further refinement. The SUMO algorithm works by presenting hypothesized modularization to the user, who will agree with some relations, and disagree with others. The developer's corrections can be integrated into the modularization process, in turn leading to a new modularization, which can again be refined. SUMO algorithm takes an existing partition of the system Mod – this might be the current directory structure of the system, or a modularization proposed by existing tools such as Bunch as input. It uses Solve function and Identify correction function. It is possible to iteratively feed domain knowledge into the re-modularization process. Although it does not rely upon clustering techniques, it can benefit from them in the sense that outputs from existing tools can be used as starting points for further refinement. These are some of the advantages of this paper.

In [10] (Gabriele Bavota et al. 2012) proposes the use of Interactive Genetic Algorithms (IGAs) to integrate, into a re-modularization approach, a mechanism allowing developers to feed-back automatically produced re-modularizations. IGAs are a variant of Genetic Algorithms (GAs) in which the fitness function is partially or entirely evaluated by a human while the GA evolves. The basic idea of the Interactive GA (IGA) is to periodically add a constraint to the GA such that some specific components shall be put in a given cluster among those created so far. Thus, the IGA evolves exactly as the non-interactive GA. Then, every nGens generations, the best individual is selected and shown to the developer. Then, the developer analyzes the proposed solutions and provides feedback indicating that certain components shall be moved from a cluster to another. After enacting the developer's indications, a new GA population is created from such a best solution, and then the GA evolves for further nGens generations, keeping into account the provided constraints. The critical point is how to guide the developer to provide feedback. In general, one could ask developer all possible kind of feedback. However, this would make the developer's task more difficult. For this reason, two different kinds of IGAs are proposed in order to guide developers in providing feedback. The first one referred to as Random-Interactive Genetic Algorithm(R-IGA) takes the best solution produced by the GA, randomly selects two components (from the same cluster or from different clusters), and then asks the developer whether, in the new solutions to be generated, such components must be placed in the same cluster (i.e., stay together) or whether they should be kept separated. Isolated Cluster (IC-IGA)is the second variant of IGA used in guiding the developer in providing the feedback. It focuses on specific parts of the re-modularization produced by the GA. Automatic re-modularization approach often tend to create a large number of small clusters. Therefore, the second variant of IGA asks feedback on the nClusters smallest clusters in the best solution (in terms of MQ).Then, for each of these clusters, if it is an isolated cluster then the developer is asked to specify a different cluster where the isolated component must be placed while for not isolated clusters, the developer is asked to specify for each pair of components whether they must stay together or not.

5. CONCLUSION

This paper gives a survey on various re-modularization techniques that are widely used for software system re-modularization. Many techniques like hierarchical clustering, automated clustering techniques and clustering using Genetic Algorithm and Interactive GA are discussed. For the survey it is concluded that the usage of Interactive Genetic Algorithm is more efficient in re-modularizing a software system because it considers the feedback from the developer while re-modularizing the software system.

6. ACKNOWLEDGEMENT

I would like to express my deepest gratitude and warming appreciation to Mr. R. Bright Gee Varghese, Assistant Professor, Department of Computer Science and Engineering, Karunya University for his guidance and I would like to thank authors of all the references which helped to make this survey meaningful.

7. REFERENCES

[1] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional (1999).

[2] Maqbool, O., Babri, H.A.: Hierarchical clustering for software architecture recovery. IEEE TSE 33(11), 759{780 (2007).

[3] Maqbool, O., Babri, H.A.: Automated software clustering:An insight using cluster labels.The Journalof Systems and Software 79(2006) 1632-1648.

[4] Hani Abdeen, Stephen Ducasse, Houari Sahraoui, Ilham Alloui: Automatic Package Coupling and Cycle Minimization. Author manuscript, published in ”The working Conference on Reverse Engineering(WCRE) (2009).

[5] Doval, D., Mancoridis, S., Mitchell, B.S.: Automatic clustering of software systems using a genetic algorithm. In: STEP. pp. 73-82. (1999)

[6] Tonella, P., Susi, A., Palma, F.: Using interactive GA for requirements prioritization. In: SSBSE. pp. 57-66 (2010)

[7] Bavota, G., Lucia, A.D., Marcus, A., Oliveto, R.: Software re-modularization based on structural and semantic metrics. In: WCRE. pp. 195{204. (2010)

[8] Dirk Beyer, Andreas Noack :Clustering Software Artifacts Based on Frequent Common Changes, proceedings of the 13th International Workshop on Program Comprehension (IWPC’05) 1092-8138/05 IEEE.

[9] Mathew Hall, Neil Walkinshaw, Phil McMinn : Supervised Software Modularization, presentation at 28th IEEE International Conference on Software Maintenance, 23-30 September 2012.

[10] Gabriele Bavota, Filomena Carnevale:Putting the developer in a loop:an interactive GA for Software Re-modularization. Search based software engineering, Lecture notes in computer science volume 7515,pp 75-89 (2012).

BIOGRAPHIES

Rajalakshmi M is pursuing M.Tech (Computer Science and Engineering) degree from Karunya University, Tamil Nadu, India. She received her B.Tech (Information Technology) degree from RVS College of Engineering and Technology, Tamil Nadu, India.

Bright Gee Varghese. R completed his B.E (Computer Science and Engineering) from Sun College of Engineering and Technology, Nagercoil, Tamil Nadu, India in 2003. He started his career as Lecturer in Sun College of Engineering and Technology, Nagercoil, from June 2003. He completed his M.E from Vinayaka Mission university, Salem, Tamil Nadu in 2011.Currently he is working as an Assistant Professor in Computer Science Department in Karunya University and pursuing part time Ph.D in Karunya University, Tamil Nadu, India. His main research area is Software Engineering.