IOSR Journal of Engineering
May. 2012, Vol. 2(5) pp: 1064-1068

Component identification from existing object oriented system using Hierarchical clustering

Shivani Budhkar[1], Dr. Arpita Gopal[2]
[1](Modern College of Engineering, Pune-05, India)
2(Sinhgad Institute of Business Administration and Research, Pune, India)

ABSTRACT
One of the important tasks in software engineering is software architecture modeling. High level software architecture is useful in all phases of software life cycle. Component based software architecture is beneficial as it is useful for reusing system parts represented as components. Most of the existing object- oriented systems do not have reliable software architecture as system evolves. To deal with this problem we have proposed approach of architecture recovery which aims to extract component based architecture from existing object oriented system using existing dependencies among classes and Agglomerative Hierarchical clustering Algorithm. In this paper, we will identify components from existing object oriented system. The tool has been developed for this purpose. We have evaluated the feasibility of this tool on Java software.

Keywords: Agglomerative hierarchical clustering, component-based, component identification, dependency, object oriented

1. INTRODUCTION
Computing environments are evolving to distributed systems where Object-oriented development had not provided extensive reuse. Component-based software architecture is a high level abstraction of a system. It has architectural elements: components which provide functionality, connectors which describe interactions and configuration which represents the topology of connections between components. This abstraction provides many advantages in the software life cycle like better abstraction capabilities, better flexibility for evolution and maintenance, better reusability as compared to object oriented paradigm. Hence, it is important to extract component based architecture from an object oriented system. Such architectures gives better understanding of legacy object oriented code as stated in [1] and identified components can be packaged, integrated into component libraries for further reuse in other new applications [2].
When software architecture is recovered, different abstraction levels can be considered for example method level, variable level, object level and system level. Extensive literature research has justified these abstraction levels for software measures [3]. The definition of metrics on Object Oriented system elements are obtained by identification of different types of relationship between different classes and computation of their strengths [4]. Class coupling is one of the Object Oriented metric. Coupling is an indication of the connections between elements of the object oriented Design [5] and indicates dependencies among classes. It is important to identify coupling for creating components. The possible dependencies among Object Oriented system entities include inheritance, composition, aggregation and method invocations. So for identifying these dependencies, the first step to identify components is performed [6].Then we used these dependencies and Agglomerative hierarchical clustering algorithm to create components in component based system. The main advantage of this approach is automation level which decreases the need of human expertise which is expensive and is not always available.
Thus, this paper presents a tool and process for identifying components using existing dependencies among classes and agglomerative hierarchical algorithm. We have evaluated our approach on small java program. Four components were identified.
The rest of the paper is organized as follows: In section 2 we propose a tool and process for identifying components. Section 3 presents overview of clustering, similarity measure, and algorithm. Case study and results from our tool are provided in section 4. Section 5 gives related work. Section 6 concludes and proposes idea for future

2. TOOL AND PROCESS
We have identified three steps to produce a component based architectural view from an object-oriented application in our approach i) Identify dependencies in existing object oriented system ii) Identify components iii) Identify the provided and required interfaces to bind them together. Figure 1 shows overall approach for producing component based architecture.

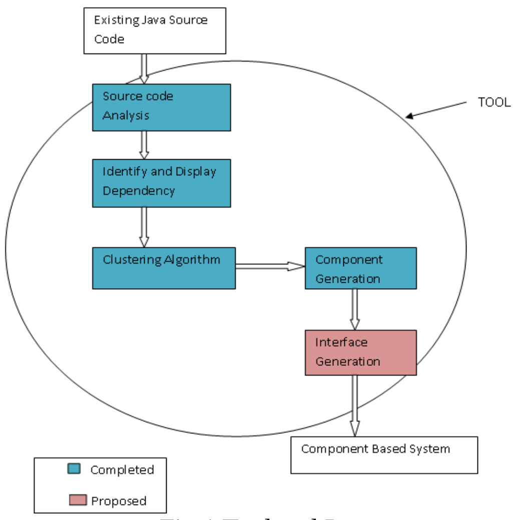

Fig.1 Tool and Process

i) Identify dependencies in existing object oriented system: Our process is based on the identification of source code entities and the relationship between them. The list of possible relationships between object oriented systems includes inheritance, composition, invocation relationship etc. [6]
ii) Identify components: - A component is group of classes collaborating to provide a function of application [7].We need to group the classes based on similarity to generate component based system from existing object oriented system. Each of the group becomes component. A hierarchical clustering algorithm allows grouping of classes of the application.
iii) Identify the provided and required interfaces: - Identified group of classes working together will form components. We also need to identify required and provided interfaces to describe how they bind together.
A tool is being developed tool which will accept the user input of an existing java source code and then generates dependencies. The tool analyzes data represented through these dependencies. These are further taken as an input to Agglomerative clustering algorithm which creates components for component based system. Similarity distance function is defined and threshold is decided.

Identify Components:-
Step i) above - Identify dependencies from existing object oriented system has been already completed [6] which generates output as integrated coupling of classes, using which cohesion measure distance d(si,sj) is formed and given input to component identification step. Cohesion measure distance d(si,sj) gives distance between two classes si and sj of object oriented system S. More details about this are given in section 3.1.Component identification step uses Agglomerative hierarchical algorithm presented in section 3.2 which helps to group similar classes together into a cluster. Finally using cluster levels components are created which can be used for creating interfaces of the components. Figure -2 summarizes our process in the tool of identifying components.

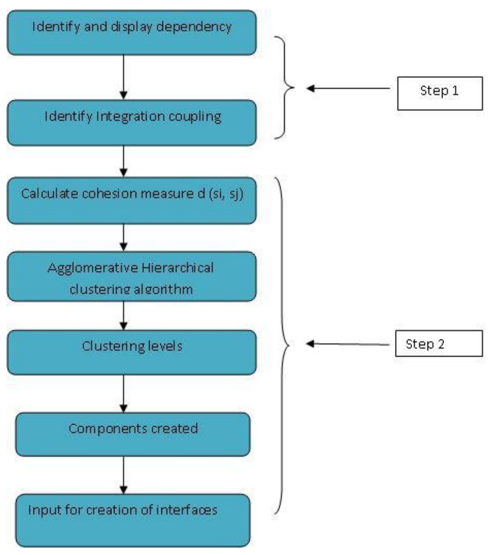
Identify and display dependency
Identify Integration coupling
Calculate cohesion measure d (si, sj)
Agglomerative Hierarchical clustering algorithm
Clustering levels
Components created
Input for creation of interfaces
Step 1
Step 2

Fig.-2 Process for identifying components

3. OVERVIEW OF CLUSTERING

Clustering is most important unsupervised learning problem. It is data mining activity for differentiating groups (classes or clusters) inside given set of objects so that objects within clusters have high similarity in comparison to one another but are very dissimilar to objects in other clusters [8]. The important thing in clustering process is the notion of degree of similarity (or dissimilarity) between the objects. If o= $\{o_1,o_2,o_3,\ldots\ldots..o_n\}$ is the set of objects to be clustered then the measure used for discriminating objects can be any metric function or semi-metric function $d{:}o\times o \rightarrow R$. The distance between two objects shows dissimilarity between them.
Various clustering algorithms are available in the literature [8][9].Most popular clustering algorithms are based on partitional and hierarchical clustering methods. Here we are focusing on hierarchical clustering method. There are two types of hierarchical clustering algorithms: agglomerative (bottom-up) and divisive (top-down). In this paper, we are using agglomerative hierarchical clustering process. For agglomerative hierarchical clustering, given a set of n objects, this algorithm begins with n singletons i.e. sets with one element, merging them until a single cluster is reached. As classes are fundamental units of object oriented system and we are reengineering object oriented legacy code into component based system, classes are used as entities to be clustered.
The agglomerative clustering algorithms differ in the way two most similar clusters are calculated and the linkage metric used. The linkage metric are single linkage, complete linkage, average linkage. The single link algorithms merge the clusters whose distance between their closest objects is the smallest. Complete linkage algorithms merge the clusters whose distance between their most distant objects is the smallest. Average link algorithms merge the clusters whose average distance i.e. the average of distances between the objects from the clusters is smallest [8][9].In this paper we are using single linkage algorithm.

3.1 Similarity measure and distance function:-

The most important factor in clustering process is similarity measure. Similarity measures determine how similar a pair of classes is. Similarity of classes can be calculated by variety of ways and choosing similarity measure is influence the result than the algorithm.
We will adapt the generic cohesion measure introduced in [3] that is connected with theory of similarity and dissimilarity. Hence cohesion measure is appropriate for our approach. We consider distance d(si, sj) between two classes si and sj from S is expressed in the following expression (1) where S= {s1,s2,……..,Sn} be the set of objects to be clustered. Objective here is to group similar classes from S in order to obtain high cohesive groups (clusters).

$$d(si,sj) = 1\text{-}sim(si,sj) \;\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Where,

$$sim(si,sj) = \frac{|\ b(si) \cap b(sj)\ |}{|\ b(si) \cup b(sj)\ |}$$

With b (si):= {Pi ЄB| si possess Pi},Pi – set of relevant properties of si.

So, distance measure focuses on the similarity measure of two entities with respect to a property subset B shown above. The distance function d(si,sj) is normalized between 0 and 1.We have chosen distance between two classes as expressed in equation (1) because it emphasizes idea of cohesion. "Cohesion refers to the degree to which module components belong together"[3]. So the equation (1) highlights concepts of cohesion. d is a semi-metric function so hierarchical clustering algorithm can be applied.

Most commonly used distances in object oriented system are distance measured through method coupling i.e. usage relationship, distance measured through composition coupling and distance measured through inheritance coupling. We have proposed integrated coupling of these three couplings [6] and used as distance measure, for Agglomerative clustering algorithm.

3.2 Algorithm:-

Software system is composed of set of classes and dependencies among the classes. The semi metric function d(si,sj) is calculated using existing dependencies in object oriented system, which is one of the important input to the clustering process. The clustering algorithm is shown as below.

```
Inputs: - The object oriented software system S= {s1, s2, s3...sn} where s1, s2,...sn are classes of object oriented
System and n is number of classes, the threshold chosen is 0.7 and the semi-metric function d between entities.
Output:-clusters at different level
Algorithm :
P=n;   //Initial number of clusters
For i=1 to n
Ci = {si}
End for
C = {c1, c2,...cp} // clusters in the system
Repeat
        d(ci,cj)=1-sim(ci,cj)  // Evaluate all pair wise distances between clusters
        -Construct distance matrix using distance value (using entity e Є S, p(e)- a set of relevant properties of e)
        -look for the pair of clusters with shortest distance.
        -Remove the pair from the matrix and merge them.
        -Evaluate all distances from this new cluster to all other clusters and update the matrix.
Until      the distance matrix is reduced to a single element.
```

4. CASE STUDY AND RESULT

We have developed tool for migration from object oriented system to component based system. Java source code is input to our tool for parsing. It shows inheritance coupling, method coupling and composition coupling and integrated coupling from java source code. Results showed most of the classes are placed in proper coupling tables [6]. We used small java software 'Arithmetic24 Game', which is developed in Java by Huahai Yang [10] as a case study. It is a simulation of popular traditional card game. Then using the integrated coupling table we calculated semi- metric function d(si,sj) for software system S. The function d is normalized between 0 and 1. So the threshold chosen is 0.7 for similarity. Using these inputs to our clustering algorithm, we got cluster levels from 0 to 4 which are shown in the figure 3. Using cluster levels components are created for 'Arithmetic24 Game'. We have identified four components for the same, as shown in figure 4.

We identified 20 classes of 'Arithmetic24 game. We have compared the result with the class diagram generated with the tool Enterprise Architecture [11].Figure 5 shows these 20 classes are placed in four components by our tool. Also several quality metrics for evaluation are proposed in literature. In [12] Cui and Chae proposed size as evaluation criteria to show well organized components with appropriate number of implementation classes. So using size we evaluate clustering results. According to them sum of ratios of single class component, classes in largest component and other intermediate components should be 100%.

Ratio of Single class component=Number of Single class component/Total number of classes

Ratio of classes in largest component=Number of classes in the largest component/Total number of classes

Ratio of other intermediate components = Number of classes in intermediate components /Total number of classes

Consider our case study,'Arithmetic24 game' which composed of 20 classes. From figure 5 the largest components are component0 and component3 consisting of 8 classes each. So Ratio of classes in largest component0 =8/20 = 40% and Ratio of classes in largest component3 =8/20 = 40%. There is a single class component, component2, so Ratio of Single class component=1/20 = 5%.There is one intermediate component, component1, so Ratio of other intermediate components = 3/20 = 15%. Thus sum of these three ratios is 100%, it indicates all the classes in the software have been considered by three ratios. Hence we say that our tool gives optimum results for component identification.

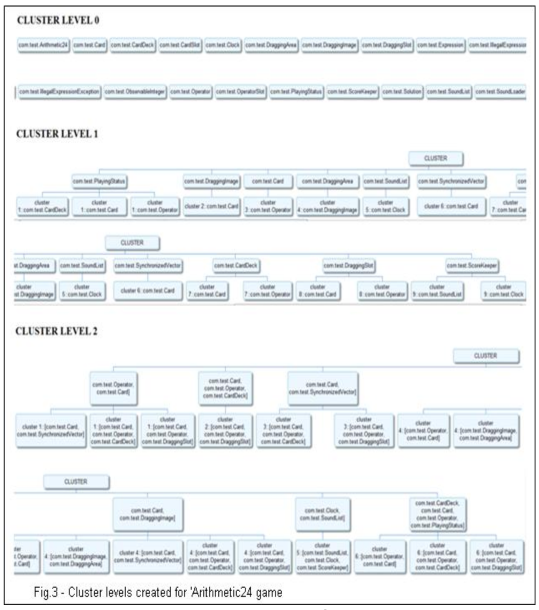

Fig.3 – Cluster levels created for ‘Arithmetic24’ game

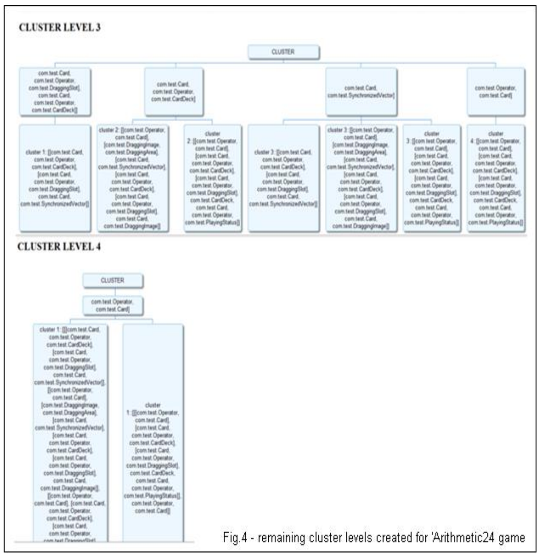
CLUSTER LEVEL 3
CLUSTER LEVEL 4
Fig.4 - remaining cluster levels created for 'Arithmetic24 game

Fig.4 –remaining cluster levels created for Arithmetic24’ game

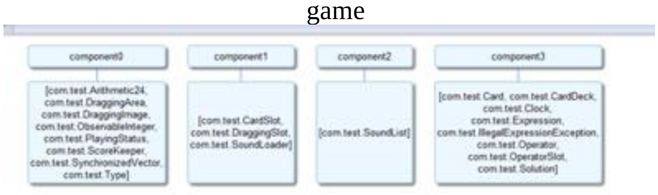
component0
[com.test.Arithmetic24, com.test.DraggingArea, com.test.DraggingImage, com.test.ObservableInteger, com.test.PlayingStatus, com.test.ScoreKeeper, com.test.SynchronizedVector, com.test.Type]
component1
[com.test.CardSlot, com.test.DraggingSlot, com.test.SoundLoader]
component2
[com.test.SoundList]
component3
[com.test.Card, com.test.CardDeck, com.test.Clock, com.test.Expression, com.test.IllegalExpressionException, com.test.Operator, com.test.OperatorSlot, com.test.Solution]

Fig.-5 components created for ‘Arithmetic24 game’

5. RELATED WORK

Medvidovic, [1] proposed Focus, a guideline to a hybrid process which regroups classes and maps the extracted entities to a conceptual architecture obtained from an architectural style according to the human expertise. Chardigny,et al proposed ROMANTIC [13] approach which is quasi-automatic. Similar to our work Alae- Eddin et al recovered component based Architecture via relational concept analysis [4]. Using Annealing simulation algorithm and concept of lattice Eunjoo Lee et.al presented a reengineering process of migrating existing object oriented system into components that are domain specific functional units [16]. Jong Kook Lee et.al proposed a component identification method that considers class cohesion, class interaction coupling ,class static coupling[14].Hassan Mathkour et al. demonstrates the generation of component based system from object oriented Design that has been achieved by developing a system recovery tool[15]. Simon Allier et.al developed automatic approach for migration from object oriented to component based system which uses Execution traces to extract data and uses clustering algorithm for component identification [7]. Suk Kyung Shin and Soo Dong Kim proposed techniques for transforming Object Oriented Design into Component Based Design using Object-Z specification. Also proposes set of rules for transforming Object Oriented Design to Component Based Design [17]. Ghulam Rasool et al. conducted case study on six different types of software systems having source code in different programming languages using the architectural recovery framework [18].

6. CONCLUSION AND FUTURE WORK

We have developed a tool which accepts object oriented java source code and migrates into component based system. The tool identifies, inheritance coupling, composition coupling, method coupling as well as integrated coupling of among classes. Using integrated coupling, tool calculates distance matrix for Agglomerative hierarchical clustering algorithm. Cluster levels created are used in creating components. We used this tool for ‘Arithmetic24 game’ written in Java and showed it is applicable to object oriented system. The tool has successfully extracted four components.

Our future work will focus on defining provided and required interfaces for the components created by tool. Once interfaces are created, we will demonstrate further evaluation for coupling and cohesion of component based system. Finally further evaluation on larger and more complex programs is needed to assess how methodology scales to deal with real industrial scale.

REFERENCES

[1] Medvidovic, N., Jakobac, V.: Using software evolution to focus architectural recovery. *Automated Software Eng. 13(2)*, 225–256 (2006)

[2] Hironori Washizaki and Yoshiaki Fukazawa.A technique for automatic component extraction from object oriented programs by refactoring. *Sci. Comput. Program,56(1-2)*:99-116,2005

[3] Frank Simon ,Silvio Loffler, Claus Lewerentz. Al Distance based cohesion measuring, Accepted for *FESMA99, Amsterdam* 4. –8. October

[4] Alae-Eddine El Hamdouni, A.Djamel Seriai1, and Marianne Huchard Component based architecture recovery From OO systems via relational Concept Analysis,*CLA10 7th International Conference on Concept Lattices and Their Applications, Sevilla : Spain (2010)*pg-259-270

[5] Roger S. Pressman, *Software Engineering A practitioner’s Approach Sixth Edition*(McGraw Hill Publications)

[6] Shivani Budhkar and Dr.Arpita gopal Component Based Architecture recovery from object oriented system using existing dependencies, *International Journal of Computational Intelligence Techniques ,Volume 3, Issue 1, 2012*, pp.-56-59.

[7] Simon Allier Salah Sadou,Houari Sahraoui and Regis Fleurquin, From Object-Oriented Applications to Component–Oriented Applications via Component oriented Architecture, *Ninth Working IEEE/IFIP Conference on Software Architecture,2011* pg- 214-223

[8] Jiawei Han and Micheline Kamber, *Data Mining Concepts and Techniques second edition(* Elsevier publisher,2006)

[9] Jain A.M,Murthy M.N and Flynn P.J. Dam, Clustering: A review: *ACM Computing surveys,31(3)*;264-323,1999

[10] http://javaboutique.internet.com/arith24/

[11] Shivani Budhkar,Dr. Arpita Gopal, Reverse Engineering Java Code to Class Diagram: An Experience Report, *International Journal of Computer Applications (0975 – 8887) Volume 29– No.6, September 2011* pg. 36-43

[12] Jian Feng Cui,Heung Seok Chae, Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems, *Information and Software technology 53(2011)*601-614]

[13] Chardigny, S., Seriai, A., Oussalah, M., Tamzalit, D.: Extraction of component-based architecture from object-oriented systems. In: *WICSA. pp. 285–288. IEEE Computer Society(2008)*

[14] Jong kook Lee, Seung Jae Jung,Soo Dong Kim, Woo Hyun Jang,Dong Han Ham, “Component Identification Method with Coupling and cohesion” in proceedings of *Eight Asia pacific Software Engineering Conference(APSEC’01)* 1530-1362/01

[15] Hassan Mathkour, Ameur Touir,Hind Hakami,Ghazy Assassa,On the Transformation of Object oriented-based systems to Component Based systems,*IEEE international conference on Signal Image technology and Internet Based systems,2008* pg- 11-15

[16] Eunjoo Lee,Byungjeong Lee,Woochang Shin, Chisu,”A reverse engineering Process for Migrating from object oriented Legacy system to a component based system”,*In proceedings of 27th Annual International Computer Software and Applications Conference 2003*, 0730-3157/03

[17] Suk Kyung Shin and Soo Dong Kim A Method to Transform Object Oriented Design into Component-Based Design using Object-Z. *The third ACIS international conference on Software Engineering Research, Management and Applications (SERA’05)* pg-274 - 281

[18] Ghulam Rasool and Naddim Asif Software Architecture Recovery, *International Journal of Computer, Information and Systems, Science and Engineering Summer 2007*, pg 99- 104