Utilizing Graph Matching for Mining Design Patterns

Ayushi Jain, Aditya Verma, Kumud Pant, Akshara Pande

Abstract: Design Patterns are one of the demonstrated reusable answers for the normally experienced design issues. The identification of design pattern is significant action that underpins re-building procedure and gives insights to the designer. The uncovering of these design patterns help understand the object oriented models clearly by analyzing the relations present in the model. Many design pattern identification approaches have been proposed in past years. These methodologies work upon the behavioral, structural and semantic analysis of the software. Many algorithms were used to recognize design patterns in software. In this paper, we will be extracting an attribute relational matrix from the graph using object oriented approach. The aim of the paper is to discover all the design patterns present in the system design.

Keywords: Patterns, algorithms, semantic.

I. INTRODUCTION

There is a need of hour to develop design patterns because of its usability. Design patterns are used to make designs more flexible, elegant and robust. Detection of design patterns will be the first step in the software system as it helps to understand and manage system software efficiently.There are 23 proposed design patterns (DP) by Gang of-Four (GoF). Mining of these GoF DPs aid to software developers for better understanding the design of the code.

This section provides us the knowledge of different methods that were involved in detecting the patterns in system design. The design patterns that have to be identified can be structural, creational or behavioral.

Brown[5] was the primary scientist who put forth an attempt towards the automatic detection of design patterns. The researcher reverse-engineered the Smalltalk code to facilitate the recovery of GOF design patterns provided by Gamma.Design pattern instances can be recovered by database queries which were proposed by Rasool et al. [9].

Vincent D. Blondel[2] has proposed a technique to figure the comparability of two vertices. In recent years, graph theory played a vital role to solve applications in real as well as practical life. It has additionally been utilized for mining of configuration designs.

Nikolas Tsantalis et al. [1] presented comparability scoring algorithm for detection of patterns. This method only computes the similarity between vertices of graph, so the fundamental disadvantage of this algorithm is that it can't compute the comparability between graphs. Jing Dong [11]proposed Template Matching method by the help of this we can compute the similarity insubgraphs.

Revised Manuscript Received on August 10, 2019.

Ayushi Jain, Graphic Era Hill University, Dehradun, India.
Aditya Verma, Graphic Era Hill University, Dehradun, India.
Kumud Pant, Graphic Era Deemed to be University, Dehradun, India.
Akshara Pande, Graphic Era Hill University, Dehradun, India. E-mail: pandeakshara@gmail.com

Another strategy for detection of design patterns is Difference figuring technique which works on UML models [8] created by S. Wenzel. The upsides of S. Wenzel approach over other discovery strategy is that it can recover incomplete instances of design patterns additionally. Bergenti and Poggi[4] proposed a procedure of designpattern discovery.

Fuzzy graph approach for configuration design discovery and Klenberg approach utilized before for detection of design patterns. This approach has a restriction that they are not engaged about entire diagram and just centered on similarity of vertices. To resolve the above restrictions many sub graph matching algorithm is proposed for identification. Another system DNIT (Depth-Node-Input Table) [6] is created for identification of design patterns, which helps to find out patterns which present at various positions of rooted directed graphs.

The another work [7]based on graph matching.The aim is to recognize the available design patterns in system design. The upside of this methodology over other structure design discovery approaches was that the memory necessity was very lower and it identifies presence of each design pattern.

The various approaches[3] concentrated on measurements for example, association, dependency, aggregation and so forth of system design. The measurement approach of pattern identification is utilized to limit the searching effectively.

Uchiyama et al.[10] proposed a method which uses source code metric and AI for recovery of design patterns. Numerous devices are utilized for recovery of design patterns. These tools help to identify design patterns from source code.

Gupta et al. [12] utilized the Normalized cross relationship (NCC) for the detection of patterns. They began by composing the relationship networks of the graphs by extracting the UMLdiagrams. Further, they applied the NCC to find the level of closeness between the two graphs, in this case they correspond to system design and design pattern.

The paper presents graph matching technique “KMP-pattern recognition algorithmfor detecting the design pattern in source code. Knuth Morris Pratt (KMP) is a searching algorithm, which search for a given pattern from the string.The principal thought behind KMP's algorithm is, at whatever point we get a mismatch (after specific matches), we unquestionably know a segment of the characters in the text of the window. By utilizing this information we may stay away the text that we know will match.The benefit of utilizing KMP is that it doesn't backtracks in a text string which helps in a type of capacity.

The applied KMP algorithm enhances the worst case complexity to O(N+M). The complexity of this algorithm for a matrix N*N of length M will be O(N*(N+M)). The time taken for each row and column is O(N+M).

II. BACKGROUND AND MOTIVATION

Matrix allow us to create large- scale attribute structure without disrupting or destroying the structure of the system design or design pattern. Matrices are used to retain the values that are Maintenance is one of the most crucial and costly stage of the development in a software. Design Patterns makes the interaction between designers efficiently by isolating the variability that are present in the system environment, thus, makes the system easy to understand and maintain. In this paper, we are discussing how a matrix could constitute a particular pattern and system design pattern definitions.

A. Different representations corresponding to Source code and Design Pattern

UML diagrams can be transformed into directed graphs which represent the relationships between the components in the graph. Let DPG:(V, E) where V represents vertex set and an edge set E and SPG:(V',E') contains vertex set V' and edge set E' of the design pattern and system design respectively. Nodes can be labeled using variables and edges can be labeled using relationship between the nodes. The relationships can be denoted by "agg" for aggregation, gen for generalization, and so on. We can create graphs for all the UML diagrams.

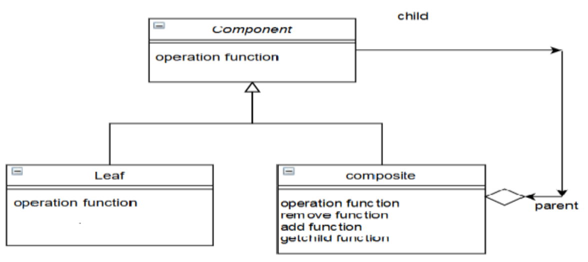

Fig.1 Composite Design pattern UML diagram

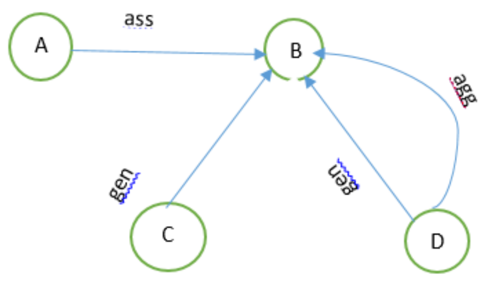

Fig.2 Graph representation of Composite design pattern

The obtained graphs are then used to create the adjacency matrices for DPG and SPG as DPA[V][V] i.e. Design pattern Adjacency and SPA[V][V] i.e. System pattern Adjacency, where V is the number of nodes present in a graph. An adjacency matrix is a square matrix which indicates whether the vertices pairs are adjacent or not. If the value at DPA[A][B] is 1, then there exist a relationship between the vertices that means there exist an edge between the two vertices in the graph and vice versa. The relations in the graphs are categorized and the matrix is filled accordingly. Similarly, matrices can be formed for system design pattern.

The Adjacency Matrix for the relations is as follows:

Association Generalization Aggregation

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The algorithm searches for the design pattern adjacency matrix in the system software pattern. The edges in DPA should be identical to the edges in SPA i.e. the edges adjacent to vertex α in DPA and vertex α' in SPA are equivalent if they share the same label and the directions respectively. The incoming edges and outgoing edges at α in DPA and α'in SPA should be same or the degree of α is less than degree of α'. The labels on the edges for same vertex in both the matrix are same. The attribute relation matrix for system software design is calculated in the similar way for all the attributes present in the pattern.

Thus, the algorithm maps one by one, all the values in the design pattern to the system design. This is done by comparing each edge in the matrix for a particular relation to the edge in another matrix. With the help of this process, a mapping of some edges may be obtained and identify a full design pattern in the system design. But, if some edges do not get mapped and full design does not get mapped then the mapping is cancelled.

III. DESIGN PATTERN DETECTION USING KMP

1. Extract the UML diagram from the design pattern and the system pattern.
2. Extract the matrices corresponding to UML for the system pattern and the design pattern as SPA and DPA respectively.
3. Apply KMP to detect the occurrence of design pattern in the system pattern.

Algorithm DPD_KMP:

1. Select vertices from DPA as V1, V2 with connecting edge as E1.
2. Mine the system pattern for the similar occurrence of E1 using KMP.
3. Call KMP function.

The design patterns in the system pattern is detected using KMP:

i. Set the initial position to 0 i.e (pos=0).

ii. Check whether position of current element is less than the length of DPA.

   a. If the condition is true, then check whether (SPA[k]=DPA[j]) then increment the value of j and k by 1.

   b. Again check whether (k=len(SPA)), if condition is true then display “PATTERN FOUND”.

   c. Set the position in DPA at which an element is found to j-k. Again increment position to 1, then set k=T[k].

iii. The condition at step ii. isfalse and set the value of k=T[k].

   a. Then check whether k<0 and increment value of j and k to 1.

IV. CONCLUSION

This paper has been designed to find the design pattern in the system pattern using KMP. It uses matrices for comparing the graphs. The paper efficiently matches the relation attribute in the design pattern to the system pattern. The applied approach can be used to identify other relations present in the system pattern in a similar way.

REFERENCES

1. N. Tsantalis, A.Chatzigeorgiou, G. Stephanides, and S. Halkidis, “Design Pattern Detection Using Similarity Scoring,” IEEEtransaction on software engineering, 32(11), 2006.
2. V.D. Blondel, A. Gajardo, M. Heymans, P. Senellart,, and P.VanDooren, A Measure of Similarity between Graph Vertices:Applications to Synonym Extraction and Web Searching, SIAM Rev., vol. 46, no. 4, pp. 647-666, 2004.
3. H. Bunke and B. T. Messmer.“ Recent advances in graph matching. Int. J. Pattern Recognition and Artificial Intelligence,11(1):169{203, 1997.
4. F. Bergenti and A. Poggi, Improving UML Designs Using Automatic Design Pattern Detection, Proc. 12th Int‟l Conf. Software Eng. and Knowledge Eng.(SEKE‟00), July 2000.
5. K. Brown,“Design Reverse-Engineering and Automated Design Pattern in Smalltalk”, Technical Report TR-96-07,Dept. Of Computer Science,North Carolina State Univ.,1996.
6. A. Pande, M.. Gupta, A.K. Tripathi “DNIT – A New Approach for Design Pattern Detection”, International Conference onComputer and Communication Technology, MNNIT- Allahabad, proceeding published by the IEEE, 2010.
7. M. Gupta, R.R. Singh, A. Pande, A.K.Tripathi,“Design pattern Mining Using State Space Representation of Graph Matching”,1st International Conference on Computer Science and Information Technology, Banglore, 2011, to be published by LNCS,Springer.
8. S. Wenjel, U. Kelter, “Model-driven design pattern Detection using difference calculation method”, In Proc. of the 1stInternational Workshop on Pattern Detection for Reverse Engineering(DPD4RE), Benevento, Italy,2006.
9. G. Rasool, I. Philippow, P. Mader, “Design Pattern Recovery Based on Annotations”. International Journal of advances in Engineering Software, Vol 41, Issue 4, 2010, pp. 519- 526.
10. Uchiyama, S., Kubo, A., Washizaki, H., and Fukazawa, Y. (2014). Detecting Design Patterns in Object-Oriented Program Source Code by Using Metrics and Machine Learning. Journal of Software Engineering and Applications, 7, 983-998. doi:10.4236/jsea.2014.712086.
11. Jing Dong, Yongtao Sun, Yajing Zhao, “Design Pattern Detection by Template Matching”, the Proceedings of The 23rd Annual ACM Symposium on Applied Computing (SAC),pages 765-769, Ceará, Brazil, March 2008.
12. M. Gupta, A. Pande, R. Singh Rao, and A.K Tripathi, "Design Pattern Detection by normalized cross correlation," in Proc. IEEE InternationalConference on Methods and Models in Computer Science (ICM2CS), India, 2010, pp. 81-84.