Volume 3, Issue 11, November 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Design Pattern Detection by a Heuristic Graph Comparison Algorithm

Rajwant Singh Rao *

Department of Computer Science &Information Technology (CSIT)
Guru Ghasidas Vishwavidyalaya Bilaspur (C.G.) India

Manjari Gupta

Department Computer Science
Banaras Hindu University, Varanasi India

Abstract— Design Patterns are proven solution to common recurring design problems. Design Pattern Detection is most important activity that may support a lot to re-engineering process and thus gives significant information to the designer. Knowledge of design pattern exists in the system design improves the program understanding and software maintenance. Therefore, an automatic and reliable design pattern discovery is required. Graph theoretic approaches have been used for design pattern detection in past. Here we are applying an algorithm which is based on the heuristic graph matching. The same algorithm we are here using for design pattern detection from the system design.

Keywords— design pattern, UML, matching, one-one correspondence, matrix.

I. INTRODUCTION

Graph based approached have been used in many software engineering problems. Design Patterns are proven solutions for common recurring software design problems. The design patterns have been extensively used by software industry to reuse the design knowledge [1]. During maintenance of a software system the necessary tasks are to understand and modify it. It would be helpful to discover pattern instances in it, if any. Many algorithms have been proposed for design patterns detection like [2, 3, 4, 5]. Similar works on design pattern detection have been discussed in section II. This paper presents a design pattern detection technique by heuristic graph matching algorithm. Here, the graphs are corresponding to the relationship graphs which exist in the UML diagrams of system design (model graph or system under study) as well as in UML diagrams of design patterns. In the classic concept of exact graph matching, the aim is to determine whether two graphs are the same or whether a subgraph of one exists in the other.

The algorithm is based on the one-one correspondence from nodes of the design pattern graphs (DPG) to system design graph (model graph MG) technique. Two graph are said to match when there is a one-one correspondence from DPG to MG. The outline of this paper is as follows. In section II related works are discussed. Section III explains the representation of model graph and design patterns in terms of relationship graphs is explained. The graph matching algorithm is described in section IV. In section V the design pattern detection is described using some examples. Lastly we concluded in section VI.

II. RELATED WORK

The first attempt for automatically detecting design pattern was by Brown [6]. In this work, Smalltalk code was reverse-engineered to facilitate the detection of four well-known patterns from the catalog by Gamma et al. [1]. Antoniol et al. [5] developed a technique to identify structural patterns in a system to observe how useful a design pattern recovery tool could be in program understanding and maintenance. Nikolaos Tsantalis [2] proposed a methodology for design pattern detection using similarity scoring. However, the limitation of similarity algorithm is that it only calculates the similarity between two vertices, not the similarity between two graphs. Jing Dong [3] gave another approach called template matching, which calculates the similarity between sub graphs of two graphs instead of vertices, to solve the above limitation. S. Wenzel [4] purposed a difference calculation method works on UML models. The advantage of difference calculation method on other design pattern detecting technique is that it detects the incomplete pattern instances also. Bergenti and Poggi [7] developed a method that examines UML diagrams and proposes the software architect modifications to the design that lead to design patterns. . Kim et al. Champin et al. [8] proposed a new method to recover the GoF1 patterns using software measurement skills. They developed a design pattern CASE tool to facilitate the easy application of their method. DPR method used three kinds of product metrics, and the measurement plan was established on the basis of the GQM paradigm. Many other tools have been developed for design pattern detection. But there is no standard tool for it that can be used to solve the maintainer's problem. Stencel and Wegrzynowicz, [9] proposed a method for automatic design pattern detection that is able to detect many nonstandard implementation variants of design pattern. Their method was customizable because a new pattern retrieval query can be introduced along with modifying an existing one and then repeat the detection using the results of earlier source code analysis stored in a relational database. Drawback was that the method was not general enough to identify all design patterns. Further the translation of first order logic formulae as SQL queries is very laborious and error-prone.

In our earlier work, we used klenberg approach for vertices scoring and fuzzy graph algorithms for design pattern detection [11]. But the drawback of these two methods is they are only concerned about node similarity not the whole graph. We used sub graph isomorphism detection approach that overcomes this drawback [11]. We have used these and other approaches for design pattern detection in GIS application [12]. To reduce complexity of design pattern detecting algorithm we used the graph decomposition technique [13]. The order of complexity of this decomposition algorithm is O(n3), where n is the number of nodes present in the graph. This algorithm works for only those design patterns having similar relationships among at most three classes in its UML class diagram. However this condition may not hold for only few of the design patterns. Thus this approach can be applied for almost all of the design patterns. In another work we find out whether design pattern matches to any subgraph of system design by using decision tree [14]. A decision tree is developed with the help of row-column elements, and then it is traversed to identify patterns. By applying the decision tree approach, the complexity is reduced. We proposed a new approach 'DNIT' (Depth-Node-Input Table) [15]. It is based on the concept of depths from the randomly chosen initial node (also called root node which has depth zero) in directed graph. In another work we applied state space representation of graph matching algorithm to detect design patterns [16]. State space representation easily describes the graph matching process. The advantage of this method used for design pattern detection was that the memory requirement was quite lower than from other similar algorithms. Another advantage is that it detects variants as well as any occurrence of each design patterns. Inexact graph matching [17, 18] was also used for design pattern detection. We showed that normalized cross correlation can also be used for this [19].

III. RELATIONSHIP GRAPH REPRESENTATIONS

The system under study or the system for which we have the source code is taken first, the corresponding the class diagram of UML of that code (object oriented system) is drawn. After that the relationship graphs (that exists in UML diagram) is extracted. We have taken the UML Diagram of system designs shown in Figure 1. There are three relationships (i.e. generalization, direct association and aggregation), the corresponding relationship graphs (i.e. directed graph) are shown in Figure 2. Generalization relationship graph has relationship between only four of the nodes, Direct Association relationship graph has relationship between three of the nodes and Dependency relationship between two of the nodes.

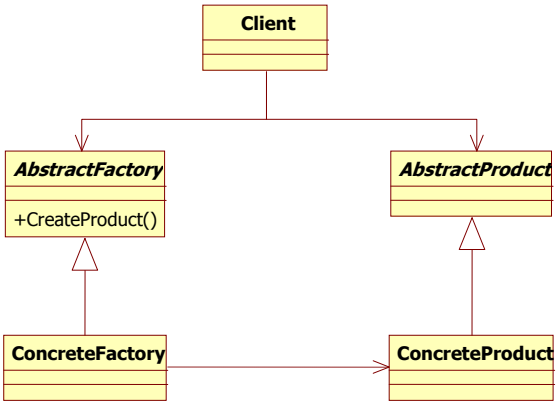
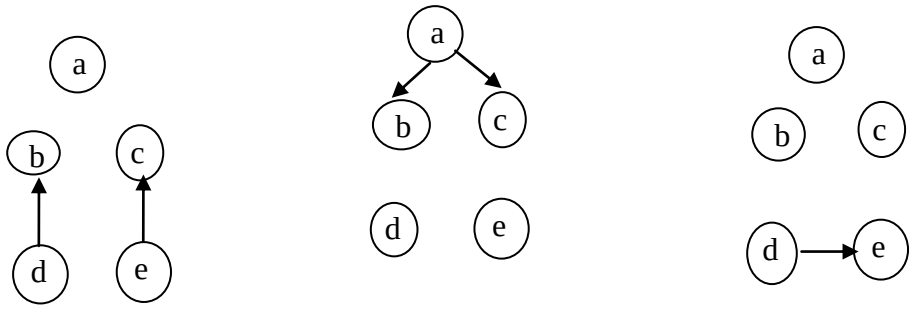


Fig. 1. UML Diagram of system design [10]



i. Generalization Graph ii. Direct Association Graph iii. Dependency Graph

Fig. 2 Corresponding Graphs for UML diagram shown in Fig. 4

IV. GRAPH MATCHING ALGORITHM

Let us consider a directed graph G (V, E), where V is a set of vertices (nodes) and E is a set of edges [20]. To compare two graphs it is necessary to identify corresponding vertices. The correspondences between two nodes a one-one correspondence [20]. Consider two graphs, G_1 (V_1, $E1$) and G_2 (V_2, $E2$), and a matrix representing the correspondences between V_1 and V_2. Here $V_1$and V_2 denote sets of vertices, in graph G_1 and G2 respectively, similarly E_1 and E_2 are the set of edges in graph G_1 and G2 respectively. Let n is the number of one-one correspondences between V_1 and V_2 [20].

In the one-one correspondence matrix we assign the $(i,j)^{th}$ entry as 0 if there will be a one-one correspondence from node 'i' to node 'j', where node 'i' belongs to DPG and node 'j' belongs to MG. if all the nodes of DPG have a one-one correspondence to some of the nodes in MG, then it will be a full exact matching. If some of the nodes of DPG have one-one correspondence in MG and some of the nodes of DPG don't have one-one correspondence then it will be partial matching, and if none of the nodes of DPG have one-one correspondence to nodes of MG then it will be not matching.

V. DESIGN PATTERN DETECTION USING GRAPH-MATCHING ALGORITHM

There are 23 GoF (Gang of Four) [1] design patterns. UML diagrams can be drawn for each of the corresponding design patterns. Here we are considering some of them.

All these cases are described in detail by using examples.

A. *Design Pattern Detection as Strategy Design Pattern: Exact Matching*

Firstly, we are considering. Factory Design Pattern, the UML diagram and corresponding relationship graph (DPG) is shown in Fig. 3 and Fig. 4 respectively. The factory design pattern has only one relationship that is direct association and its corresponding graph is shown in fig 4. Now we can make a correspondence between the nodes of fig 4 and fig (2.ii).There will be a correspondence between node '1' and node 'a' i.e. [1,a], and there will be correspondence between node '2' to 'b' and node '2' to node 'c' i.e. [2,b] and [2,c]. We can represent these correspondence by a matrix in which the $(i,j)^{th}$ entry will be 0 if there is a correspondence from any node 'i' belong to design pattern graph to a node 'j' belong to model graph. The correspondence between fig 4 to fig (2.ii) is shown in Table I . Here each node of fig 4 has a one – one correspondence in fig (2.ii) so there will be an exact matching.

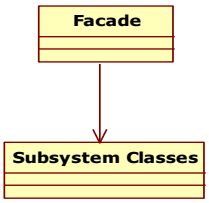


Fig 3. Factory Design Pattern [10]

Fig 4.Direct Association Graph of Fig 3

TABLE I . Direct association one- one matrix

	a	b	c	d	e
1	0				
2		0	0		

B. *Design Pattern Detection as Command Design Pattern: Partial Matching*

In some cases it is also possible that a particular design pattern partially exist in the system design pattern (case ii discussed in section 4). For example consider the Mediator Design pattern, the UML diagram and corresponding relationship graph (DPG) is shown in Fig. 5 and Fig. 6 respectively. In this case we will not find the full matching.

Here in the generalization one-one relationship matrix (Table II) only the nodes '1' and '2' have the one-one correspondence, but the node '3' don't have any one – one correspondence. So it's a partial matching. Similarly we can see that in direct association one-one matrix (Table III) only the nodes '1' and '2' have one-one mapping, but node '3' don't have any correspodence, so its partial matching found.

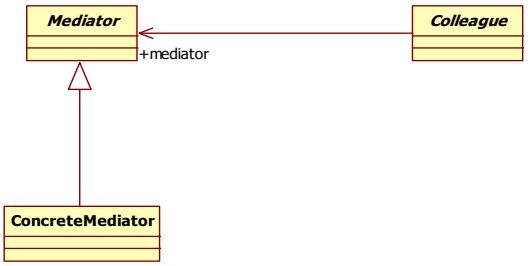


Fig 5.Mediator Design Pattern [10]

i. Generalization graph ii. Direct Association Graph

Fig. 6 Corresponding Graphs for UML diagram shown in Fig. 6

TABLE III. Generalization one- one matrix

	a	b	c	D	e
1		0	0		
2				0	0
3					

TABLE IIIII Direct Association one-one matrix

	a	b	c	D	e
1		0	0		
2					
3	0				

C. *Particular design pattern may not exist*

Above we have seen the examples of design pattern existence (complete or partially) but it can be possible that a particular design pattern does not exist in the model graph. In this case we will not find any matching between relationship matrices. For example if we take singleton design pattern (Fig. 7), there is only one relationship: direct association on itself node. Corresponding DPG is shown in Fig. 8 and its Direct Association one -one matrix shown in Table IV. Here there is no one-one correspondence from node '1' to any of the nodes belonging in model graph nodes. So matching is not found.

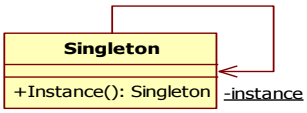


Fig.7 Singleton Design pattern [10]

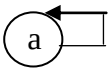


Fig 8. Graph for fig 11

TABLE IVV Direct Association one -one matrix

	a	b	c	d	e
1					

VI. CONCLUSIONS

This paper presents an approach for design pattern detection using A Heuristic Graph Comparison Algorithm technique. We took the relationship graphs of the model graph (MG) and a design pattern (DPG), after that the corresponding relationship one-one correspondence matrices are created and then and tried to find whether all of the nodes of DPG have a one-one correspondence to the nodes of MG. If all of the nodes of DPG have a one –one mapping into MG, we say that the design pattern exist in the model graph. If some of the nodes of DPG have one –one correspondence and some of the nodes don't have one-one correspondence, design pattern partially exist in the model graph, and if none of the nodes of DPG have one-one mapping to nodes of MG, design pattern does not exist in the model graph.

REFERENCES

[1]. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns Elements of Reusable Object-Oriented Software. Addison- Wesley, Reading (1995)

[2]. Tsantalis N., Chatzigeorgiou A., Stephanides G., Halkidis S., "Design Pattern Detection Using Similarity Scoring", *IEEE transaction on software engineering*, 32(11), 2006.

[3]. Dong J., Sun Y., Zhao Y., "Design Pattern Detection by Template Matching", *the Proceedings of The 23rd Annual ACM Symposium on Applied Computing (SAC)*, pages 765-769, Ceará, Brazil, 2008.

[4]. Wenzel S., Kelter U., "Model-driven design pattern Detection using difference calculation", *In Proc. of the 1st International Workshop on Pattern Detection for Reverse Engineering* (DPD4RE), Benevento, Italy, 2006.

[5]. Antoniol G., Casazza G., Di Penta M., Fiutem R., "Object-Oriented Design Patterns Recovery", *J. Systems and Software*, vol. 59, no. 2, pp. 181-196, 2001.

[6]. Brown, K.: Design Reverse-Engineering and Automated Design Pattern in Smalltalk.Technical Report TR-96-07, Dept. of Computer Science, North Carolina State Univ.(1996).

[7]. Bergenti, F., Poggi, A.: Improving UML Designs Using Automatic Design Pattern Detection. In: Proc. 12th Int'l Conf. Software Eng. and Knowledge Eng. SEKE 2000 (2000).

[8]. Champin P. A., Solnon C., "Measuring the similarity of labeled graphs", *5th International Conference on Case-Based Reasoning (ICCBR)*, Lecture Notes in computer Science- Springer Verlag, 2003.

[9]. Stencel K. and Wegrzynowicz P., "Detection of Diverse Design Pattern Variants", 15th Asia-Pacific Software Engineering Conference, IEEE Computer Society, 2008.

[10]. StarUML, The Open Source UML/MDA Platform. http://staruml.sourceforge.net/en/

[11]. Pande A., Gupta M., "Design Pattern Detection Using Graph Matching", International Journal of Computer Engineering and Information Technology (IJCEIT), Vol 15, No 20, Special Edition, pp. 59-64, 2010.

[12]. Pande A. & Gupta M., "Design Pattern Mining for GIS Application using Graph Matching Techniques", 3rd IEEE International Conference on Computer Science and Information Technology. pp. 09-11, Chengdu, China, 2010.

[13]. Pande A., Gupta M., Tripathi A.K., "A New Approach for Detecting Design Patterns by Graph Decomposition and Graph Isomorphism", International Conference on Contemporary Computing, Jaypee Noida, CCIS, Springer, 2010.

[14]. Pande A., Gupta M., Tripathi A.K., "A Decision Tree Approach for Design Patterns Detection by Subgraph Isomorphism", International Conference on Advances in Information and Communication Technologies, ICT 2010, Kochi, Kerala, LNCS-CCIS, Springer 2010.

[15]. Pande A., Gupta M., Tripathi A.K., "DNIT – A New Approach for Design Pattern Detection", International Conference on Computer and Communication Technology, MNNIT- Allahabad, proceeding published by the IEEE, 2010.

[16]. Gupta M., Singh R.R., Pande A., Tripathi A.K., "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.

[17]. Gupta M. Singh R.R., Tripathi A.K., "Design Pattern Detection using Inexact Graph Matching", International Conference on Communication and Computational Intelligence, Tamil nadu, Dec 2010, to be published by IEEE Explore.

[18]. Gupta M., "Inexact Graph Matching for Design Pattern Detection using Genetic Algorithm", International Conference on Computer Engineering and Technology, Nov 2010, Jodhpur, to be published by IEEE Explore.

[19]. Manjari Gupta, Akshara Pande, Rajwant Singh Rao, A.K. Tripathi, Design Pattern Detection by Normalized Cross Correlation, International Conference on Methods and Models in Computer Sciences (ICM2CS-2010),December 13-14, 2010, JNU, to be published by IEEE Explore.

[20]. Hiroyuki Ogata, Wataru Fujibuchi, Susumu Goto, and Minoru Kanehisa, A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters.

