International Journal of Advanced Computer Science, Vol. 2, No. 3, Pp. 114-119, Mar. 2012.

Design Pattern Mining for GIS Application Using Graph Matching Techniques

Akshara Pande, Manjari Gupta, & A K Tripathi

Manuscript
Received: *30,Jun., 2011*
Revised: *20,Oct., 2011*
Accepted: *28,Feb., 2012*
Published: *15,Apr., 2012*

Keywords
design pattern, graph distance, UML, matrix, template matching, normalized crossed correlation, subgraph isomorphism

***Abstract*— Design Pattern Detection is a part of many solutions to Software Engineering difficulties. It is a part of reengineering process and thus gives important information to the designer. Design Pattern existence improve the program understanding and software maintenance. With the help of these patterns specific design problem can be solved and object oriented design become more flexible and reusable. Hence a reliable design pattern mining is required. A GIS is an information system designed to work with data referenced by spatial / geographical coordinates. Here we are detecting design patterns so that it can be used as a conceptual tool to cope with recurrent problems appearing in the GIS domain. In this way, GIS applications can evolve smoothly, because maintenance is achieved by focusing on different concerns at different times.**

1. Introduction

Geographic Information Systems deal with many characteristics such as data acquisition, accuracy, representation of spatial relationships, topological features and interface design The designers must consider them when developing geographic applications. In simple words, GIS is used for database applications that store and analyze georeferenced data both. The relational model upon which most Current GIS software systems are built has been acknowledged as an insufficient model for applications that deal with spatial data [1]-[2]. The use of object oriented technology was proposed for design of Geographical Information System and now it is becoming a growing trend in the GIS applications [3]-[4]. In GIS domain there are many recurrent problems involving the use of spatial information such as object locations, coordinate manipulation, computation of geographic functions, and so on.

In many object oriented software, there are recurring patterns of classes. Design Patterns are defined as explanation of corresponding classes that forms a common solution to frequent design problem. To reuse expert design experiences, the design patterns [5] have been extensively used by software industry. A software design pattern, filter and refine, which is widely used in spatial database design to improve the efficiency of spatial retrieval. When patterns are implemented in a system, the pattern-related information is generally no longer available. It is tough to trace out such design information. To understand the systems and to modifications in them it is necessary to recover pattern instances. There are number of pattern detection techniques [6]-[8]. In this paper we firstly draw the equivalent UML diagram for GIS application and then try to find out whether a particular design pattern exists in that application or not by applying graph matching techniques [9]-[12]. In this paper we propose methods (i.e. Graph Distance Approach, Normalized Crossed Correlation and Subgraph Isomorphism Detection) for design pattern detection. The detailed methods are given in below sections.

2. UML Diagram in GIS Domain

Consider a problem in which the area is selected by GIS system which consists of buildings (like schools, hospitals, residents, offices etc), empty lots and blocks.

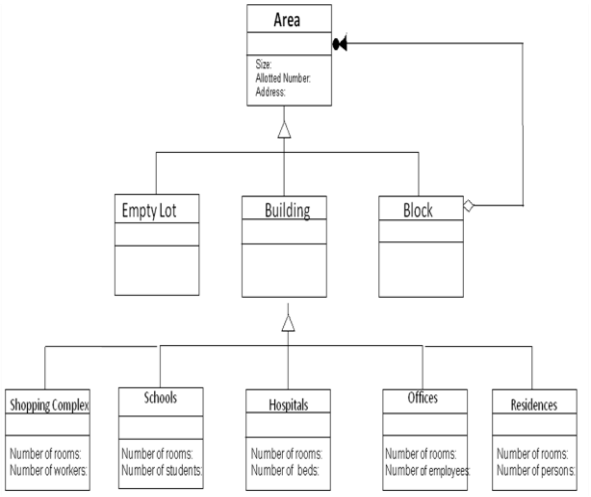


Fig. 1. UML diagram s of an Area which is developed under GIS

3. Graph Matching Techniques

Graph Matching techniques are important and very general form of pattern matching that finds realistic use in areas such as image processing, pattern recognition and computer vision, graph grammars, graph transformation, bio

computing, search operation in chemical structural formulae database, etc.

Graph Distance

Let g1 = (V1, E1, α1, β1) and g2 = (V2, E2, α2, β2) be graphs. A common subgraph of g1 and g2, CS (g1, g2), is a graph g (V, E, α, β). We call g a minimum common subgraph of g1 and g2, MCS (g1, g2) [13].The graph distance is given as [12]

$$\delta(g1,g2) = 1 - \left|MCS(g1,g2)\right| / \max(\left|g1\right|,\left|g2\right|) \quad \text{(Equ. 1)}$$

Equation (1) measures the graph distance. For any three graphs the following relations hold [12]:

- $0 \le \delta(g1,g2) \le 1$
- $\delta(g1,g2) = 0 \Leftrightarrow g1 = g2$
- $\delta(g1,g2) = \delta(g2,g1)$
- $\delta(g1,g3) \le \delta(g1,g2) + \delta(g2,g3)$

Template Matching by Normalized Crossed Correlation

Normalized cross correlation (NCC) has been used extensively for many machine vision applications. Normalized cross correlation (NCC) has been commonly used as a metric to evaluate the degree of similarity (or dissimilarity) between two compared images [10]. Here we are taking two graphs one corresponding to system and another for design pattern (template graph), then we are applying NCC to find match between template r (i, j) of size m x n and system matrix f(x, y) of size M x N. NCC is defined as

$$\delta(x,y) = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} f(x+i,y+j).r(i,j) - m.n.\mu_f.\mu_r}{\sqrt{\{(\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} f^2(x+i,y+j) - m.n.\mu_f^2).(\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} r^2(i,j) - m.n.\mu_r^2)\}}} \quad \text{(Equ. 2)}$$

For all (x ,y)€ M x N,

Where,

$$\mu_f(x,y) = \frac{1}{m.n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} f(x+i,y+j)$$

$$\mu_r(x,y) = \frac{1}{m.n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} r(i,j)$$

Subgraph Isomorphism Detection

The subgraph isomorphism is an important generalization of graph isomorphism. The subgraph isomorphism problem [14] is to determine whether a graph is isomorphic to a subgraph of another graph. Subgraph isomorphism is NP-complete.

Let [9] G1 (V1, E1) and G2 (V2, E2) be two graphs, where V1, V2 are the set of vertices and E1, E2 are the set of edges. Let M1 and M2 be the adjacency matrices corresponding to G1 and G2 respectively. A permutation matrix is a square (0, 1)-matrix that has exactly one entry 1 in each row and each column and 0's elsewhere. Two graphs G1 ($M1, L_v, L_e$) and G2 ($M2, L_v, L_e$) are said to be isomorphic [9] if there exist a permutation matrix P such that

$$M2 = P\ M1\ P^T \quad \text{(Equ. 3)}$$

Given an n x n matrix $M = (m_{ij})$, let $S_{k,m}(M)$ denote the k x m matrix that is obtained from M by deleting rows k + 1 , . . . , n and columns m + 1 , . . . , n, where k, m < n. A subgraph *S* of a graph *G*, $S \subseteq G$, is a graph $S = (M^i, L_v, L_e)$ where $M^i = S_{m,m}(P\ M\ P^T)$ is an m x m adjacency matrix for some permutation matrix P. The concept of subgraph isomorphism can now be described as follows.

Let G1 and G2 be graphs with adjacency matrices M1 and M2 of dimensions m x m and n x n respectively, where m < =n. There is a subgraph isomorphism [9] from G1 to G2 if there exists an n x n permutation matrix P such that

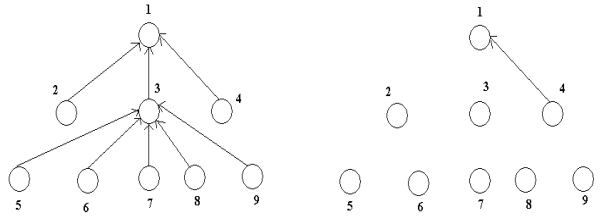
$$M1 = S_{m,m}(P\ M2\ P^T) \quad \text{(Equ. 4)}$$

We take M2 matrix as a system design matrix and we guess nondeterministically M1 as a design pattern matrix. And then try to find out whether M1 is subisomorphic to M2 or not or it can be easily said whether there exist design pattern in the system graph or not.

Hence the problem of finding a subgraph isomorphism from graph G1 to G2 is equivalent to finding a permutation matrix for which equation (4) holds. Thus, we generate permutation adjacency matrix of a model graph (system under study) one by one and check whether equation (4) holds or not, when it holds we stop and declare that that particular design pattern has been detected. It can be also possible that there is no design pattern exists in system graph. In this case we find no permutation matrix for which equation (4) holds.

Examples

We have taken the system under consideration (i.e. the UML diagram which is developed under GIS) as well as design pattern. The generalization and aggregation graph of system under study (i.e. corresponding to UML diagram shown Fig. 1) is shown in Fig. 2.



Generalization Graph Aggregation Graph

Fig. 2. Corresponding Graphs for UML diagram shown in Fig. 1

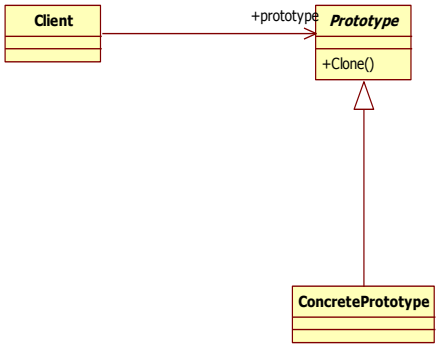
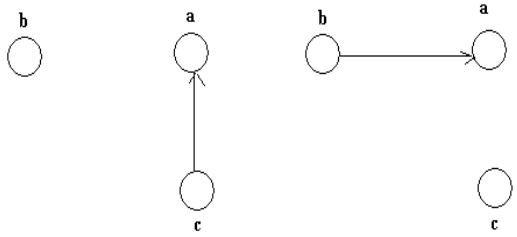


Fig. 3. Prototype Design Pattern



Generalization Graph Association Graph

Fig. 4. Corresponding graph for UML of Prototype Design Pattern

Equation (1) can be applied for calculating the graph distance between two nodes of the graph. Firstly take the generalization graph from Fig. 2 (corresponding to system design) and Fig. 4 (corresponding to Prototype design pattern). The maximum common nodes between them are 2 and maximum number of nodes is 9(for system graph). Here, $|MCS\ (g1, g2)|=2$, $\max(|g1|, |g2|)=9$, so, on applying equation (1) we have $\delta(g1, g2)$ as 7/9 i.e. 0.77.

The limitation of above method is that it only calculates the distance between two vertices rather than whole graph. To remove this drawback another approaches are present there, called normalized crossed correlation and subgraph isomorphism detection. For applying these approaches we have to consider adjacency matrices corresponding to each relationship graphs of UML diagrams.

4. Matrix Representation

Firstly we write the matrix corresponding to system design (i.e. the area which is selected by GIS system), here are two relationships for system graph, generalization and aggregation, so corresponding to them generalization matrix and aggregation matrix are shown in fig 5 and fig 6 respectively.

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0

Fig. 5. Matrix Representation of System Design corresponding to generalization

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Fig. 6. Matrix Representation of System Design corresponding to aggregation

In the similar way, generalization and association matrices for prototype design pattern are shown in figure 7 and figure 8.

	a	b	c
a	0	0	0
b	0	0	0
c	1	0	0

Fig. 7. Matrix Representation of Prototype Design Pattern corresponding to generalization

	a	b	c
a	0	0	0
b	1	0	0
c	0	0	0

Fig. 8. Matrix Representation of Prototype Design Pattern corresponding to association

Now from fig 5 and fig 7 (i.e. matrices corresponding to generalization relationships of system design and prototype design pattern), if normalized cross correlation

formula (i.e. equation 2) is applied, then the value of δ(0, 0) is approximately 0.65 (Here we are starting row from 0^{th} row instead of saying 1^{st} row). For δ(2, 2) the value is 1, which indicates the proper match means for generalization relationship the prototype design pattern exists in system design.

This is very time consuming to calculate each feature matrix (i.e. relationships present example generalization, direct association etc) separately. So we are trying to write it down in the form of overall matrix [7].

There is one or more design features are present in system and patterns, they can be represented in the form of matrices. For example there is generalization relationship, we have to create n x n matrix for n number of classes. If [7] the i^{th} and j^{th} classes have generalization relationship, the corresponding cell of the matrix should be 1, otherwise 0.

To reduce the number of manipulations, we try to combine different matrices (like generalization matrix, association matrix, dependency matrix, aggregation matrix etc) into a single matrix. So there will be only two overall matrices, one corresponding to system and one for design pattern. To combine different matrices into an overall matrix, certain root value of a prime number is given to a matrix (different feature matrices) and then combine them. The cell value [7] of each matrix is then changed to the value of its root to the power of the old cell value. Let us consider the UML diagram corresponding to system design (Fig. 1) and other UML diagrams corresponding to design pattern (i.e. Composite Design Pattern).

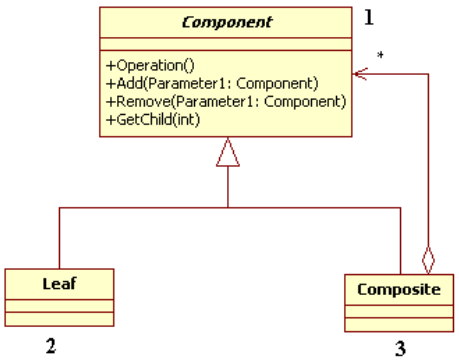


Fig. 9. Composite Design Pattern

Calculation of matrices for system design

Now firstly we calculate the overall matrix corresponding to model graph (i.e. system design Fig. 1), there are 2 corresponding matrices aggregation matrix and generalization matrix.

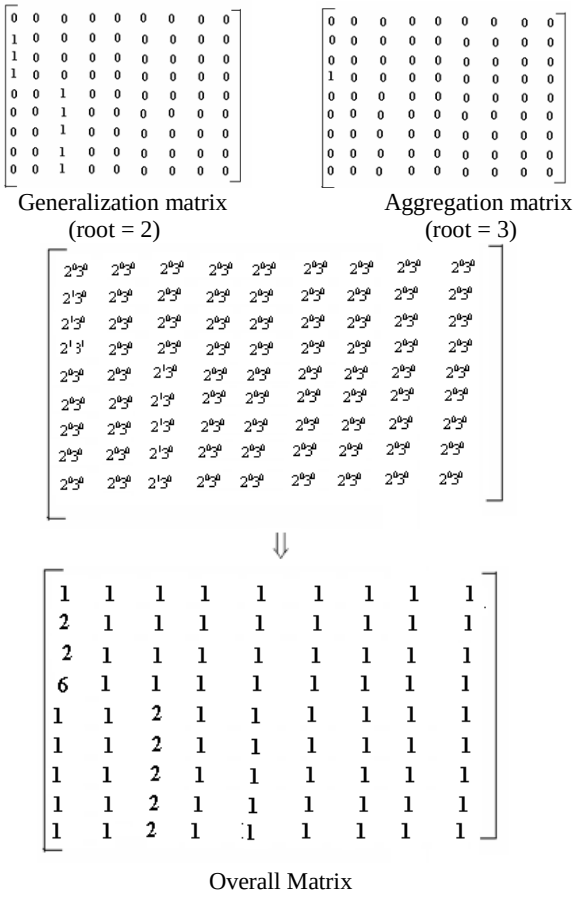


Fig. 10. Matrix Representation of System Design

Calculation of matrices for design patterns

Let us consider Composite Design Pattern (Fig. 9). There are two matrices one matrix for relationship generalization and other for aggregation relationship.

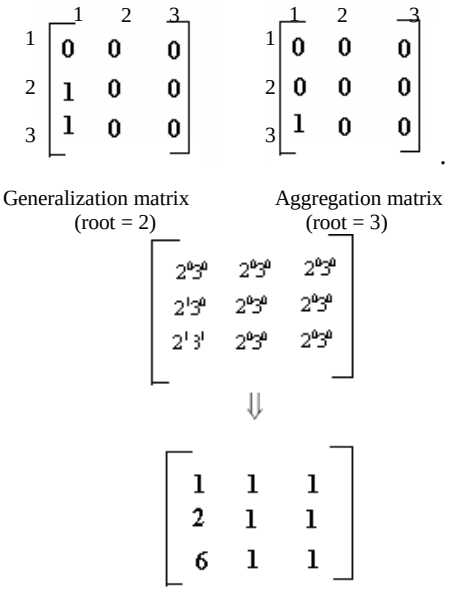


Fig. 11. Matrix Representation of Composite Design Pattern

5. Design Pattern Detection Using Subgraph Isomorphism

The main objective of this project is to detect design pattern in given system. We have UML diagrams corresponding to system design and design pattern and then we can calculate overall matrices corresponding to them (by above given procedure). Thus one matrix for system design and another for design pattern. First we non-deterministically guess any design pattern (or subgraph) of system design pattern and then by applying equation (3) it can be find out whether they really are sub-isomorphic or not or design pattern exists or not in a particular system design.

Let us consider the overall matrix corresponding to system design (Fig. 10) i.e. matrix S, and guess any subgraph of it.

Design Pattern Detection as Composite Design Pattern

First we guess that Composite Design Pattern (Fig. 9) exists in system design (or can say that there is subgraph isomorphism is in between them). The overall matrix is (Fig. 11)

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 6 & 1 & 1 \end{bmatrix}$$

Let there is a permutation matrix P of order of system design i.e. 9x9,

P =

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

P^T =

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

Calculation of P S P^T is shown below.

PSP^T =

$$\begin{bmatrix} 1&1&1&1&1&1&1&1&1 \\ 2&1&1&1&1&1&1&1&1 \\ 2&1&1&1&1&1&1&1&1 \\ 6&1&1&1&1&1&1&1&1 \\ 1&1&2&1&1&1&1&1&1 \\ 1&1&2&1&1&1&1&1&1 \\ 1&1&2&1&1&1&1&1&1 \\ 1&1&2&1&1&1&1&1&1 \\ 1&1&2&1&1&1&1&1&1 \end{bmatrix}$$

After eliminating entries from 4th row and 4th column we have reduced matrix as (because Composite Design Pattern is of 3 x 3 orders)

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$

This is not the same as Composite design pattern's overall matrix. Now if we take permutation matrix as shown below

P =

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

P^T =

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

PSP^T =

$$\begin{bmatrix} 1&1&1&1&1&1&1&1&1 \\ 2&1&1&1&1&1&1&1&1 \\ 6&1&1&1&1&1&1&1&1 \\ 2&1&1&1&1&1&1&1&1 \\ 1&2&1&1&1&1&1&1&1 \\ 1&2&1&1&1&1&1&1&1 \\ 1&2&1&1&1&1&1&1&1 \\ 1&2&1&1&1&1&1&1&1 \\ 1&2&1&1&1&1&1&1&1 \end{bmatrix}$$

After eliminating entries from 4th row and 4th column we have reduced matrix as shown below.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 6 & 1 & 1 \end{bmatrix}$$

This is the same as of Composite Design Pattern. So subpart of system design has a carbon copy as a composite design pattern in system design or can be said that Composite design pattern is detected in system design.

Particular Design Pattern may or may not exist

From above, we observe the example of design pattern existence but it can be possible that a particular design pattern does not exist in system design. In this case there will be no permutation matrix for which we can find out (after row and column elimination) a matrix which is equivalent to design pattern matrix.

6. Related Work

The first effort towards automatically detect design pattern was achieved by Brown [15]. In this work, Smalltalk code was reverse-engineered to facilitate the detection of four well-known patterns from the catalog by Gamma et al. [5].

Antoniol et al. [16] gave a technique to identify structural patterns in a system with the purpose to observe how useful a design pattern recovery tool could be in program understanding and maintenance.

Nikolaos Tsantalis [6], proposed a methodology for design pattern detection using similarity scoring. But the limitation of similarity algorithm is that it only calculates the similarity between two vertices, not the similarity between two graphs. To solve this Jing Dong [7] gave another approach called template matching, which calculates the similarity between subgraphs of two graphs instead of vertices.

S. Wenzel [17] gave the 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.

7. Conclusion

Three approaches for design pattern detection using graph matching have been discussed here in GIS domain. In this we took the model graph and a data graph (corresponding to design pattern), and tried to find out whether design pattern exists or not in model graph. There are 23 GoF (Fang of Four) [5] design patterns. The Graph Distance approach determined how much similar the two nodes are. But the drawback of this method is that it only concerned about node similarity not the whole graph. By applying normalized crossed correlation it can be determined the particular template (design pattern) exactly matches or partially matches to subpart of system design or not. Subgraph isomorphism identification technique, using overall matrix, reduces the complexity and tried to find out whether a particular design pattern exists in system design.

References

[1] T. Harder, & A. Reuter, "Architecture of Database Systems for Non-Standard Applications, (in German) In: A. Blaser and P. Pistor, editors, Database Systems in Office," (1988) *Engineering, and Scientific Environment, Springer Verlag*, New *York Lecture Notes in Computer Science*, Vol. 94.

[2] A. Frank, "Requirements for a Database Management System for a GIS," (1988) *Pho togrammetric Engineering & Remote Sensing*, vol. 54, no. 11.

[3] B.U.P. Kosters, & H.W. Six, "Object-Oriented requirements engineering for GIS applications," (1995) *inProc. of the ACM 3rd ACM International Workshop on Advances in Geographic Information Systems, ACMGIS' 95*, USA.

[4] M.A.C. Medeiros & G. Camara, "The Domus project. Building an OODB GIS for environmentalcontrol," (1994) *in Proc. of IGIS'94, International Workshop on Advanced Research in GIS, Springer Verlag LNCS*.

[5] E. Gamma, R. Helm, R. Johnson, & J. Vlissides, "Design Patterns Elements of Reusable Object-Oriented Software," (1995) *Addison- Wesley*.

[6] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, & S. Halkidis, "Design Pattern Detection Using Similarity Scoring," (2006) *IEEE transaction on software engineering*, vol. 32, no. 11.

[7] J. Dong, Y. Sun, & Y. Zhao, "Design Pattern Detection By Template Matching," (2008) *the Proceedings of the 23rd Annual ACM, Symposium on Applied Computing (SAC)*, Brazil, pp. 765-769.

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

[9] B.T. Messmer, & H. Bunke, "Subgraph isomorphism detection in polynomial time on preprocessed model graphs," (1995) *Second Asian Conference on Computer Vision*, pp. 151-155.

[10] D.M. Tsai, & Ch.T. Lin "Fast normalized cross correlation for defect detection," (2003) *Pattern Recognition Letters*, vol. 24, pp. 2625-2631.

[11] K. Bharat, & M.R. Henzinger**,** "Improved algorithms for topic distillation in a hyperlinked environment," (1998) *Proceedings of the 21st annual international ACM SIGIR Conference on Research and Development in Information Retrieval*.

[12] UM Learning, Graph Matching, (2005) *Filtering Databases of Graphs Using Machine Learning Techniques-thesis.*

[13] H. Bunke, X.Jiang, & A. Kandel, "On the Minimum Common Supergraph of two Graphs," (2000) *Computing*, vol. 65, no. 1, pp. 13-25.

[14] G. Valiente, "Algorithms On Trees And Graphs," (2002) pp. 367-380.

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

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

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