IJCSN International Journal of Computer Science and Network, Volume 3, Issue 1, February 2014
ISSN (Online) : 2277-5420
www.IJCSN.org

Design Pattern Detection by Multilayer Neural Genetic Algorithm

[1]Rajwant Singh Rao, [2]Manjari Gupta

[1] Department of Computer Science & Information Technology (CSIT), Guru Ghasidas Vishwavidyalaya, Bilaspur, Chhattis Garh (C.G.) 495009, India

[2] Department of Computer Science, Banaras Hindu University, Varanasi, Uttar Pradesh (U.P.) 221005, 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 recurrent neural network genetic algorithm for design pattern detection. The same algorithm we are here using for design pattern detection from the system design.

Keywords - *Design pattern, UML, Matching, One-one correspondence, Matrix*

1. 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 recurrent neural network genetic 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 multilayer perceptron (MLP) with genetic algorithm.

The outline of this paper is as follows. In section 2 related works are discussed. Section 3 explains the multilayer perceptron. Section 4 explain the genetic algorithm. Section 5 explain the relationship graph representation. The multilayer genetic graph matching algorithm is described in section 6. Lastly we concluded in section 7.

2. 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 GoF [1]

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].

3. Multi-Layer Perceptron (MLP) Neural Network

Typically, the network consists of a set of input source or input layers, one or more hidden or computational layers, and an output layer. The input signals are propagates through the network in a forward direction, on a layer-by-layer basis. in the MLP the error back propagation algorithm is applied which is based on error correction learning rule which consists of two passes through the different layers of the network: a forward pass and a backward pass. In forward pass the synaptic weights are fixed but in backward pass the synaptic weights are all adjusted and produce better result [26].

The three layers network with sigmoid activation function is capable to produce a decision function with enough hidden units is shown in Fig. 1.

The input layer contains the input nodes interacts with the outside environment. The input units are only buffer units that pass the signal without changing it.

The hidden layer size is left to the appreciation of the user that will estimate the number of hidden nodes by his experience or by trial and error. This number must not be too large to avoid waste of computations and slow convergence process and not too low to make the network capable to absorb the set of training pixels.

The output layer represent the number of nodes, which is equal to the number of classes, each output node representing a class [21,22,23,24,25].

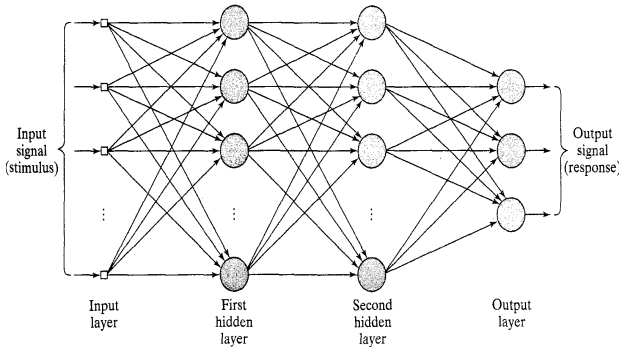

Fig.1 Multilayer Perceptron (MLP) Neural Network

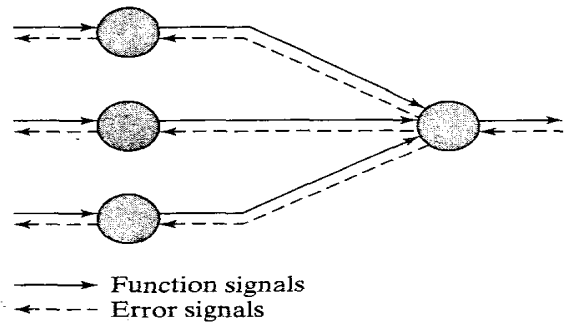

Fig. 2 Illustration of the direction of the two basic signal flows in a multilayer perceptron: forward propagation of the function signal and backward propagation of the error signal

4. Genetic Algorithm

In the GA there are parameters which need to be search. Parameters training are the range of initial weights (R), initial values of learning rate and momentum rate, and the network architecture. By using a uniform random number generator in the range [– R; R] the initial weights are generated. The number of hidden layer (h) describe the network architecture, the number of nodes in each layer (ni; i = 0; 1; : : : ; h). based on the laws of the natural selection and genetics, GA proposed as searching process. This algorithm which is used firstly to obtain the optimal values of the hidden nodes and the initial values of learning rate (R) and momentum top-level description of a simple GA is shown below [21]:

1. First of all arbitrarily create an initial population
   $P^0 = (a^0_1 \ a^0_2 \ldots a^0_n)$
2. Now calculate the fitness function F of each chromosome in the current population Pt .
3. Generate new chromosome P_{new} of mating current chromosomes, and apply the mutation and crossover process as the parent chromosomes mate.
4. Remove numbers of the population to make free room for newly created chromosomes.
5. Compute the new fitness Fnew, and insert these into population.
6. Increase number of generation, if not (completed), otherwise go to step 3, or else stop return the best chromosome.

The GA cycle is shown in Fig. 3. From which the phenotype represents the coding which is used to represent the chromosomes. The mating pool represents the summing of chromosomes that represents the solution of the problem. Offspring represents new chromosomes result by mutation and crossover operation. Lastly, the objective function represents the function that measured the performance of the origin function [21, 26, 27, 28].

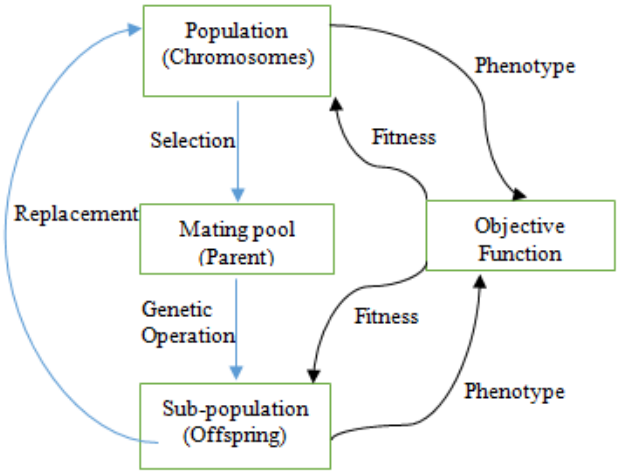

Fig 3. Genetic Algorithm Cycle.[21]

5. Relationship Graph Representation

UML diagrams of system design and design patterns are converted into graphs. Before converting a UML diagram into graphs first we modify the UML diagram in such a way so that variant of design patterns can also be detected. The reason for the design pattern variant problems is the fact that the inheritance and aggregation relationship have the property of transitiveness [2]. Thus if there is an inheritance (or aggregation) relationship between classes c1 and c2 and the same relationship between c2 and c3, we will introduce the same relationship between c1 and c3 also. Each class in the UML diagram is represented as a node and relationship is represented as an edge in the graph. If there is no relationship between any two classes, an virtual edge is created between corresponding nodes. All the nodes and edges are labeled. Each node is labeled by a 4-tuple (t_1 x t_2 x t_3 x t_4) of 0"s and 1"s, where,

t1=1 if there is an direct association relationship from the class corresponding to this node to any other class, otherwise t1=0,

t2=1 if there is an aggregation relationship from the class corresponding to this node to any other class, otherwise t2=0,

t3=1 if there is a generalization relationship from the class corresponding to this node to any other class, otherwise t3=0 and

t4=1 if there is a dependency relationship from the class corresponding to this node to any other class, otherwise t4=0.

Edges in the graph are also labeled as 4-tuple (e_1 x e_2 x e_3 x e_4) where,

e_1=1 if this edge is corresponding to direct association, otherwise e_1=0,

e_2=2 if this edge is corresponding to aggregation, otherwise e_2=0,

e_3=3 if this edge is corresponding to generalization, otherwise e_3=0 and

e_4=4 is this edge is corresponding to dependency, otherwise 0.

Since there may be more than one relationships between two classes at the same time, more than one ei"s can be non-zero for a single edge. Virtual edges will have the label (0, 0, 0, 0). The UML diagram of system design and its corresponding graph have been shown in Fig. 4 and Fig. 5 respectively. Virtual edges need to be assumed are not shown in the graph.

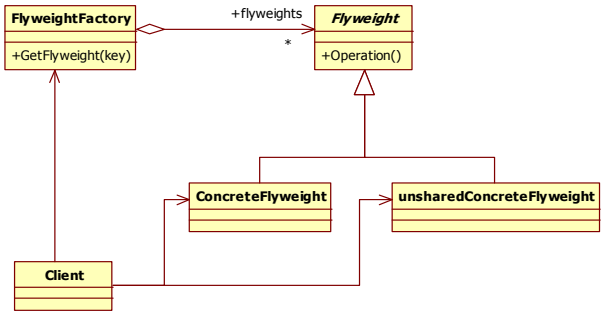

Fig. 4 UML Diagram of Flyweight Design Pattern [10]

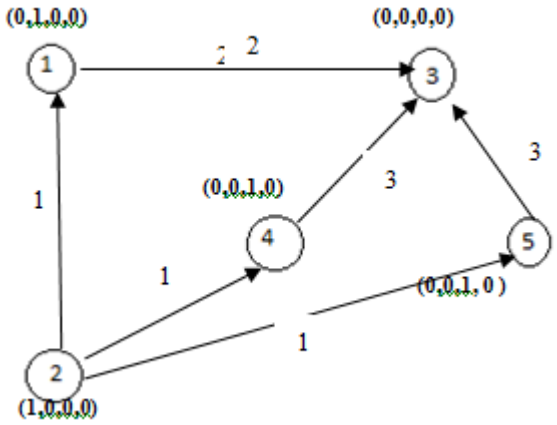

Fig. 5 Graph (MG) for Flyweight design pattern

Similarly we can represent the relationship graphs for the design patterns as for system design shown in Fig 5. For example Strategy Design Pattern and Command Design Pattern with their corresponding graphs are shown in Fig. 6, Fig. 7, Fig.8, and Fig. 9 respectively.

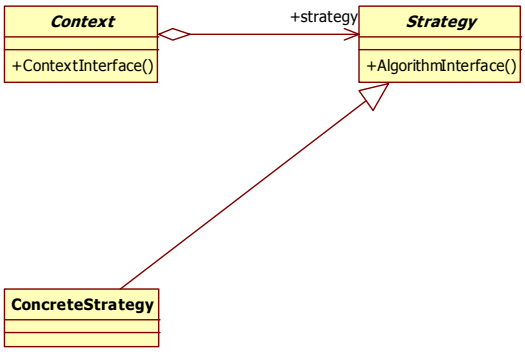

Fig 6. Strategy design pattern [10]

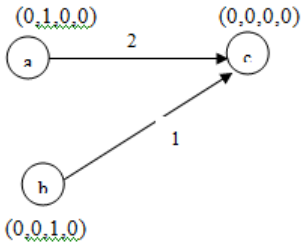

Fig. 7. Graph for strategy design pattern

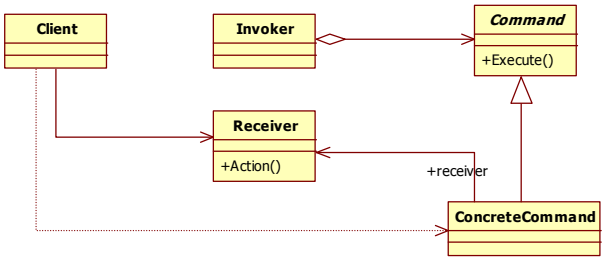

Fig 8. Command design pattern [10]

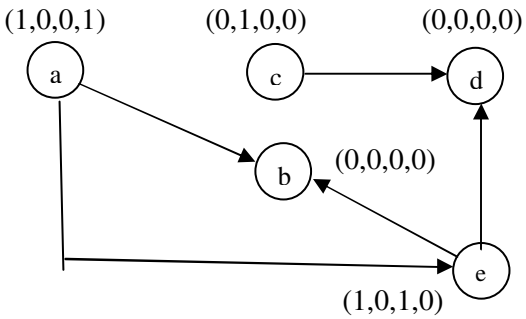

Fig 9. Graph for command design pattern

6. Multi-Layer Perceptron (MLP) Genetic Neural Network Algorithm

In system design for node to node mapping of design pattern is represented by a chromosome. In this way the obtained chromosome structure C is n x m matrix where n is the no. of nodes in design pattern graph and m is no. of nodes in system design graph. Each row and each column of C contain at most one '1' and rest of the elements are '0'. A chromosome is legal only if it represents a bijective mapping. In this the main objective is to find out chromosomes that represent sub-graph isomorphism between design pattern and system design, if any. This objective is incorporated into the fitness function of the GA. The fitness function measures the performance of the chromosomes i.e. how much the mapping represented by it is close to the sub graph isomorphism between the design pattern and the system design graph. We define the fitness function (F) as followed.

$F = F_{nc} + F_{ec}$, where F_{nc} is the cost of node mapping and F_{ec} is the cost of edge mapping. F_{nc} and F_{ec} are defined as follows:

$F_{nc} = |t_1-t'_1|+ |t_2-t'_2|+ |t_3-t'_3|+ |t_4-t'_4|$, where (t_1, t_2, t_3, t_4) and (t'_1, t'_2, t'_3, t'_4) are node labels and

$F_{ec} = |e_1-e'_1|+ |e_2-e'_2|+ |e_3-e'_3|+ |e_4-e'_4|$, where (e_1, e_2, e_3, e_4) and (e'_1, e'_2, e'_3, e'_4) are edge labels.

After generating the feasible chromosomes, using this objective function, one chromosome may be compared with another chromosome. Fundamental genetic operators: crossover, mutation and selection, for this problem, are proposed as follows. We are using 2-point crossover where crossover site will divide the rows or columns of matrix C. 1st part of (some rows or columns) one matrix is combined with 2nd part of the other matrix and we will get one new offspring. Similarly we get the other offspring by combining remaining parts of these two matrixes. Mutation randomly interchanges place of "1" (column) present in one row to the place of "1" present in other row. After applying crossover or mutation the generated offspring's may be illegal. Thus a repair function is needed to convert this illegal chromosome into a legal chromosome (that represents a bijective mapping). This repair function will check if any row or any column of a chromosome matrix contain more than one „1s", it inverts any "1" to "0" and any "0" to "1" such that each row and each column contains a single "1".

Any selection method for e.g. ranking or proportional selections can be used. The rationale is that the chromosome with lower cost function value will have a higher probability of surviving into next generation. Terminating condition may be chosen as no. of generations. After a large number of generations it will generate a mapping that will be corresponding to sub-isomorphism between design pattern and the system design graph. By backward pass the fitness function is modified and find the better result.

7. Conclusion

In this paper we proposed a new method for design pattern detection which is combination of genetic algorithm and multilayer perceptron. To detect the design pattern we took the system design and a design pattern, and tried to find out whether design pattern matches (fully or partially) to any subgraph of system design by using multilayer genetic algorithm. The encoding scheme and genetic operators: crossover, mutation and selection are discussed for solving this problem. The fitness function can be modified to improve the performance of the algorithm. The advantage of using multilayer perceptron is that we can obtain the better chromosomes for genetic algorithm and in this way we can obtain the best fitness function. We are developing a prototype that allows the implementation of the approach discussed.

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] www.myreaders.info/00-Artificial_Intelligence.pdf.

[21] Yas Abbas Alsultanny∗, Musbah M. Aqel, "Pattern recognition using multilayer neural-genetic algorithm",

[22] L. Fauselt, Fundamentals of Neural Networks, Prentice-Hall, International Inc., Englewood CliGs, NJ, 1994.

[23] R.L. Harvey, Neural Networks Principles, Prentice-Hall, Englewood CliGs, NJ, 1994.

[24] R.P. Lippmann, Introduction to computing with neural nets, IEEE Trans. Acoust. Speech Signal Process.4 (87) (1987) 3–22.

[25] J.M. Zurada, Introduction to Arti>cial Neural Systems, Jaico Pub. House, Bombay, India, 1997.

[26] D. Anthony, E. Hines, The use of genetic algorithms to learn the most appropriate inputs to neural network, Application of the International Association of Science and Technology for Development-IASTED, June, 1990, 223–226.

[27] M. Mitchill, An introduction to genetic algorithms, Massachusetts Institute of Technology, England,1996.

[28] S.c. Ng, S.H. Leung, C.Y. Chung, A. Luk, W.H. Lau, The genetic search approach, IEEE Signal process.Mag. 13 (6) (1996) 38–46.

[29] Simon haykin, Neural Networks a Comprehensive Foundation.

First Author Biographies

Mr. Rajwant Singh Rao completed his MCA (Master in Computer Application) from Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, India in 2009. Since 2009 to till date he is pursuing in PhD in Computer Science department from Banaras Hindu University, Varanasi, India. He is serving as Assistant Professor with the Department of Computer Science & Information Technology (CSIT) of Guru Ghasidas Vishwavidyalaya, Bilaspur (C.G), India since 2011. He published 3 papers in international journal. One book chapter in springer and also published/presented papers in proceeding of the international conferences.He is engaged in teaching and research for the last 4 years and the areas of his research interest include Design Pattern Detection.

Second Author Biographies

Dr. Manjari Gupta completed her master degree in Computer Science (with Second Rank) from the University of Allahabad 2002. And obtained Ph.D Degree in Computer Engineering from Institute of Technology, Banaras Hindu University in year 2006. She passed UGC-NET examination in Computer Science and Application. During her Ph.D. research, she also worked as a teaching assistant in Computer Science Section of Mahila Mahavidyalaya, Banaras Hindu University from 2003 to 2006. In September 2006, she joined Indian Institute of Information Technology at Allahabad as Lecturer in Computer Science and served there till October 2007. Thereafter she joined Banaras Hindu University Varanasi as Assistant Professor in Computer Science.. She is engaged in teaching and research for the last 10 years and the areas of her research interest include Software Reuse and Design pattern Detection. She also worked on a Project titled E-Content for Software Reuse by Design Patterns and Frameworks supported by Ministry of Human Resource and Development, Government of India.