A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm

Namita Khurana[1], Rajender Singh Chhillar[1*], Usha Chhillar[2]

[1] Department of Computer Science and Applications, M.D.U, Rohtak, Haryana, India.
[2] Department f Computer Science All India Jat P. G. College, Rohtak, Haryana, India.

* Corresponding author. Tel. :+91- 9416277507; email:Chhillar02@gmail.com
Manuscript submitted July 10, 2015; accepted December 10, 2015.
doi: 10.17706/jsw.11.3.242-250

Abstract: This paper presents a novel approach for generation of test cases from UML design diagrams. In this new generation scheme, we have considered use case diagram, activity diagram and sequence diagram. Our approach consists of converting the use case diagram into use case diagram graph (UDG), activity diagram into activity diagram graph (ADG) and sequence diagram into sequence diagram graph (SDG). After that three graphs UDG, ADG and SDG are integrated to form System Graph (SYTG). The System Graph is then traversed to generate test cases also optimized using Genetic Algorithm. The generated test cases are suitable to detect maximum number of faults like use case dependency, interaction, scenario, pre-post condition faults and error handling.

Key words: Activity diagram, genetic algorithm, online examination system, sequence diagram, system testing graph, use case diagram.

1. Introduction

Software Testing plays a very important part in life cycle of software development. It is a trade-off between budget, time and quality. With the increase in size and complexity of software products, time and effort required also increases. More than 50% of the software development cost is spent on Software testing.

Our approach for testing the object oriented system is generation of test cases from UML Models. Although it is a challenging task for analysis of UML (Unified Modeling Language) models [1] since the information is distributed across several model views yet they reduce the complexity of the problem with increase in size and complexity.

Use Case Diagrams [1] are basically for high level requirement analysis of a system. So during requirement phase of a system, functionalities are captured in use cases. Actors in the use case diagram are human users, some internal applications or may be some external applications. Fig. 1(a) shows the use case diagram for online examination system. On the other hand a sequence diagram explains how processes operate with one another and in which order. Fig. 2(a) shows the sequence diagram for Online Examination System. An Activity Diagram shows the operational work flow of any system. An activity ultimately results in some action which is shown in Fig. 3(a).

In this paper, we have proposed the test case generation from three types of UML diagrams as Use case diagram, Sequence diagram and Activity diagram. First of all use case diagram is converted in use case diagram graph, sequence diagram is converted in sequence diagram graph and activity diagram is converted in activity diagram graph. Then two algorithms are proposed for system testing graph which is formed by integrating the three graphs. After that genetic algorithm is being applied to optimize the test cases generated from system testing graph. The resulting test suite aims to cover maximum number of faults by covering all the possibilities.

The paper is organized as follows. In Section 2, we discuss the existing work done on test case generation techniques using different UML diagrams. In Section 3, we discuss the use case diagram, sequence diagram and activity diagram for online examination system. Also, we have converted the diagrams into graphs and the three are integrated into the system testing Graph (SYTG) and finally the test cases are generated and being optimized using Genetic Algorithm. Section 4 contains the Case study for test case generated and optimized for On Line Examination System. Section 5 contains the Conclusion and Future work. Last Section 6 contains the used references.

2. Related Work

Test cases are generated using three different techniques illustrated in different works like specification based, code based and model based. In this section, we survey various research papers related to test case generation techniques using UML diagrams.

Abinash Tripathy and Anirban Mitra [3] presented an approach to generate test cases by using together UML Activity diagram and Sequence Diagram [3]. In this approach first the activity diagram is being converted into activity graph and the sequence diagram is being converted into sequence graph and then the two graphs are integrated to form system Graph. Then the System Graph is being traversed to form the test cases by using Depth First Search Method (DFS) on an example of ATM card validation.

M. Sharma, Rajiv mall [4] has proposed an algorithm, to generate test case from a combination of use case diagram and sequence diagram. First of all, they convert the use case diagram into use case graph and then sequence diagram into sequence graph. Integrating the two graphs a System Graph is being generated. How the two graphs are integrated is not clearly mentioned. Also the test cases generated are not optimized.

Namita Khurana and R.S.Chhillar presented an approach [5] to Generate and optimize the test cases generated by Sequence Diagram and State Chart Diagram. First of all Sequence Diagram is being converted into Sequence Graph and the State Chart Diagram is being converted into state chart graph and then the two are integrated to form the System Graph. Then Genetic Algorithm is being applied on the resulting System Graph to generate and optimize the test cases.

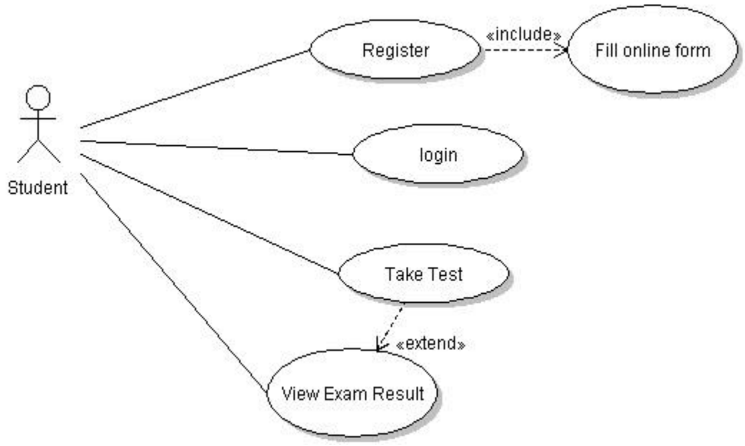
Ajay Kumar Jena, Santosh Kumar and Durga Prasad Mohapatra [6] presented an approach to generate test cases from Activity diagram. First of all an activity table is generated which is converted into Activity flow graph. After that Genetic Algorithm is applied to generate and optimize the test cases.

3. Proposed Approach

We present our work to generate optimized test cases from UML models. UML Model is standard by object management group. It is a modeling language in software engineering. It is being designed to specify, construct and document to software artifacts with support to special aspects of software such as dynamics and structural aspects [1].

3.1 Conversion of UCD to UCDG

Use case diagram also known as behavior diagrams used to describe a set of use cases (actions) that some system can perform in collaboration with one or more external users (actors).



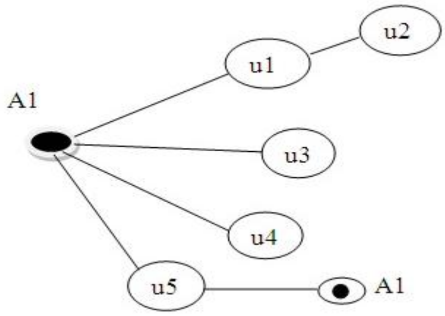


Fig. 1 (a). Use case diagram for online examination system. Fig. 1. (b). UCDG for online examination system.

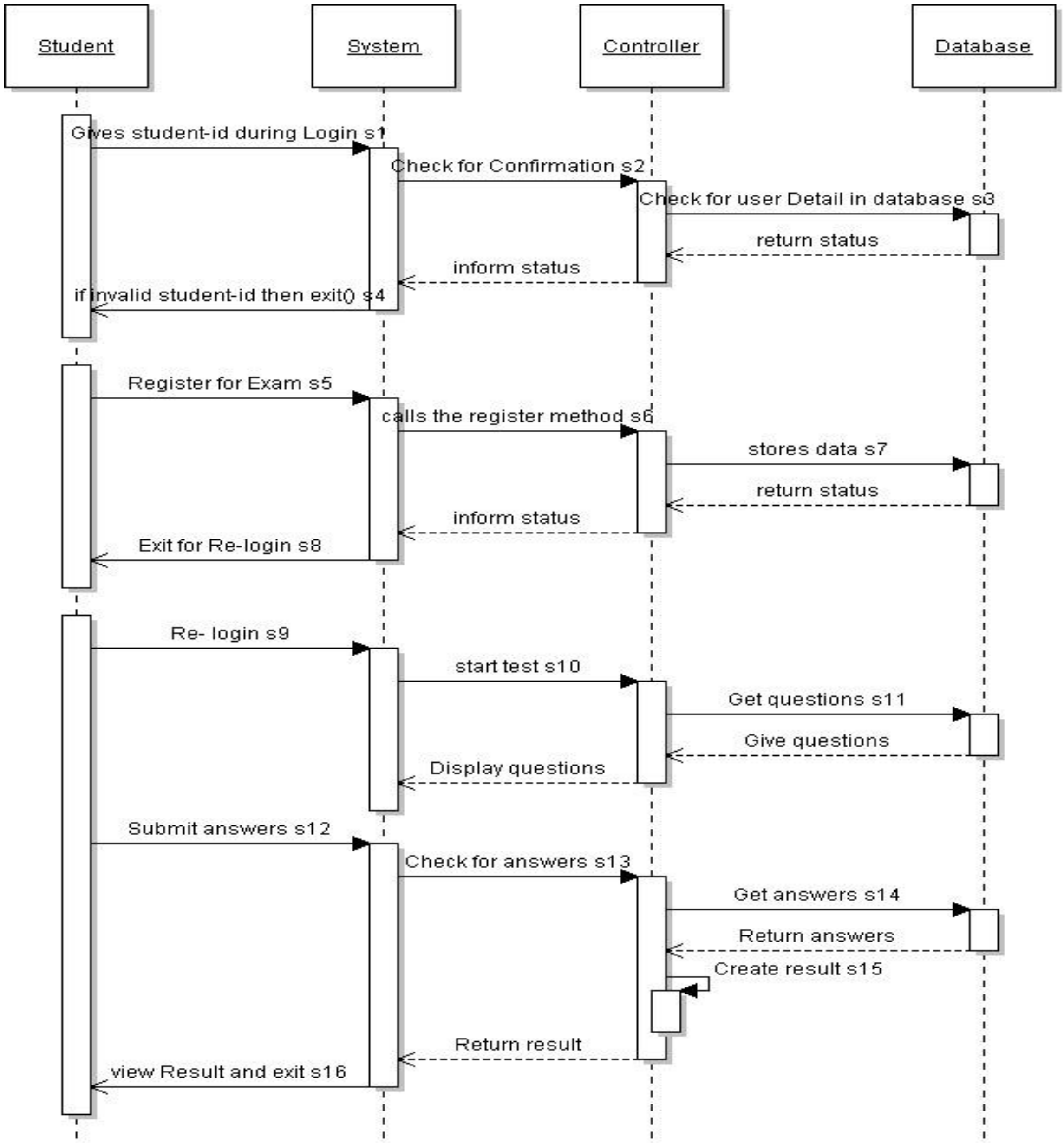


Fig. 2. (a). Sequence diagram for online examination system.

3.2 Conversion of SD to SDG

This section presents the conversion of Sequence Diagram into Sequence Diagram Graph.

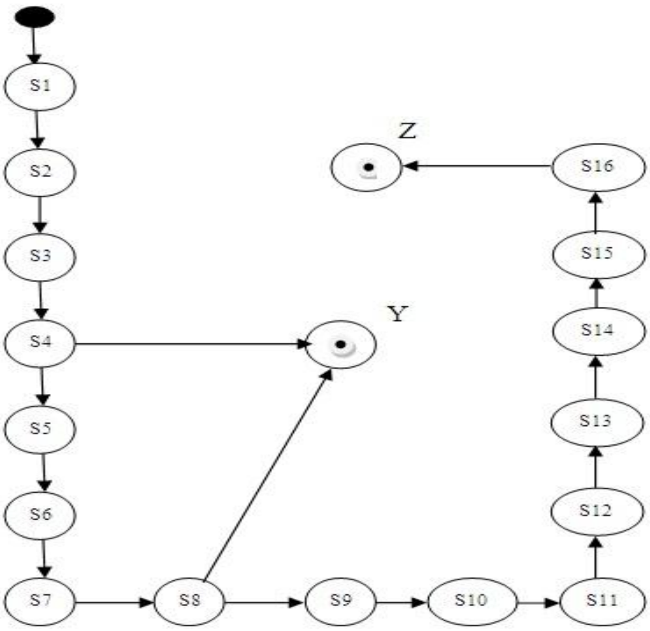


Fig. 2. (b) SDG for online examination system.

3.3 Conversion of AD into ADG

This section describes the conversion of Activity Diagram into Activity Diagram Graph.

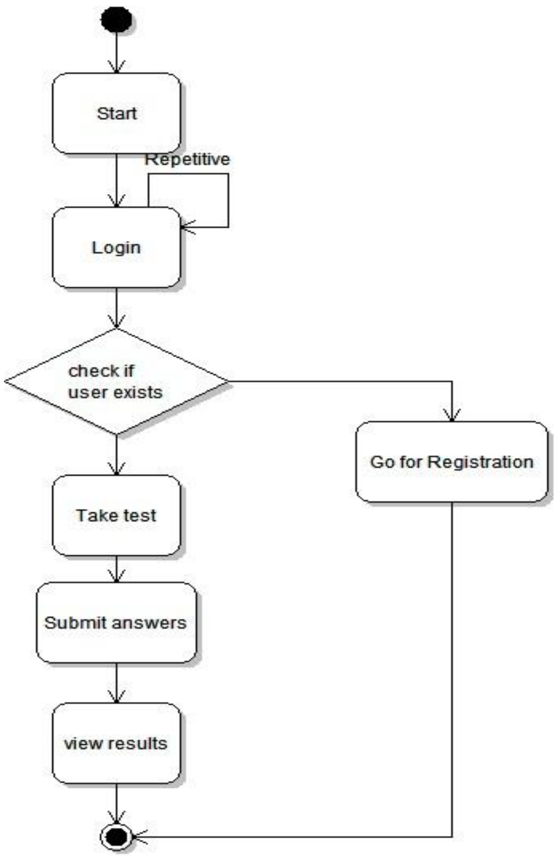


Fig. 3. (a) Activity diagram for online examination system.

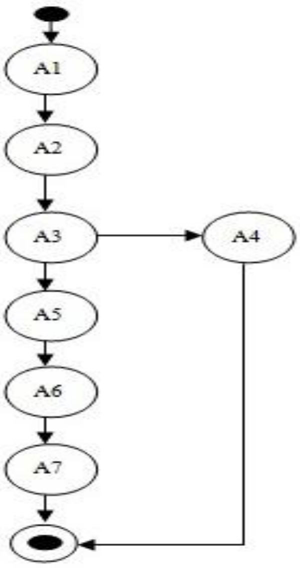


Fig. 3.(b) ADG for online examination system.

3.4 Integration of ADG,SDG and UDG into SYTG

After conversion of diagrams into Graphs, our next step is to integrate the three graphs into a single graph which is called System Graph (SYTG).

Definition of System Testing Graph (SYTG)

The system testing Graph is defined as SYTG={S,N,T,F} ,where *S* is initial node of the system graph. *N*={NUDG U NSG U NAG}is the set of all nodes of three graphs. *T*={TSG U TUDG U TAG} is set of transitions of the three graphs. *F* is the final node of SYTG.

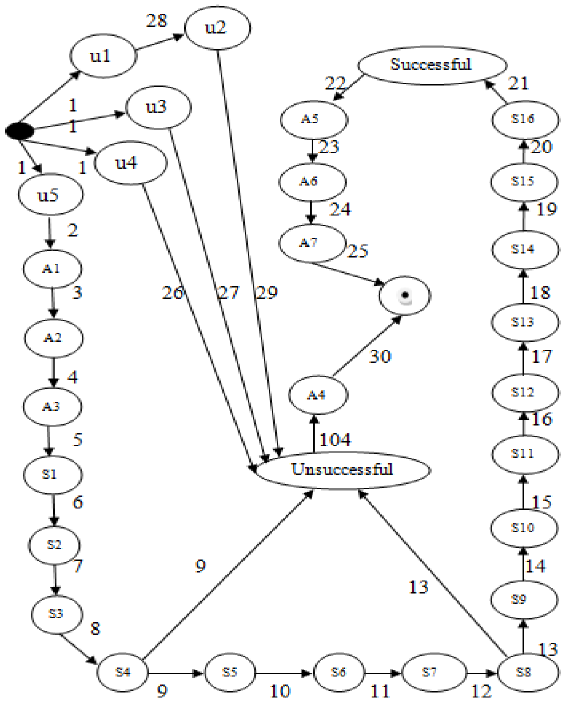


Fig. 4. SYTG graph formed after the integration of the three graphs.

Next we present the algorithm to generate the system testing graph from UCDG, SDG, and ADG.

Algorithm 1: ASDG (Activity-Sequence Diagram Graph)

Input: Activity Diagram Graph (ADG) and Sequence Diagram Graph (SDG)

Output: Activity–Sequence Diagram Graph (ASDG)

1) *P* = Identify all the paths of (ADG).
2) For each path pi ϵ *P* do
3) Aj = Ai //Start with the Ai the starting node
4) For each activity Aj of path pi do
5) If ci ϵ Aj //current node has multiple conditions
6) β = Ai-1 → SG //Edge from the previous node to the sequence Graph
7) ϒ = SG (Last) →Ai+1 //connect the last node of SG to the next node of AG. Edge from unsuccessful final node of SG to node Ai+1 where the value of V=0 else edge from successful final node of SG to node Ai+1 where the value of V=1
8) End If
9) If ci ∉ Ai
10) δ=Ai→Ai+1//connect the present node to the next node of the same Activity Graph.
11) End If
12) End For
13) End For
14) End

Algorithm 2: System Graph

Input: ASDG, UCDG

Output: System Graph (SYTG)

1) P=Identify all the paths of (UCDG)
2) For each path pi ϵ P do
3) α = ui →Ai (ASDG) //Connects the successful node of Use case diagram (UCDG) to Ai (ASDG)(Initial Node) otherwise the Unsuccessful final node of UCDG to unsuccessful node of the system Graph. Now start with ui as the starting Node.
4) End For
5) End

3.5 Generation and Optimization of Test Cases

After creating SYTG Graph, our next step is to generate the test cases. After generating the test cases we need to optimize those test cases. For Optimization we need to apply an Evolutionary Algorithm, so we apply here Genetic Algorithm for Optimization of test cases.

Algorithm 3: Test Set Generation-Optimization

Input: - System Graph (SYTG)

Output: - Optimized test cases

1) Identify all the paths P= {p1, p2, p3, p4, p5…..} from start node to a final node in SYTG.
2) Assign weights to the individual nodes. The actual weight of the child node is weight of the parent node plus one . If a child has multiple parents then weight of that node is the sum of the weights of the parent's node. Also weight is allocated to paths from left to right.
3) Calculate the (x) cost of each path as the cost of that path is sum of the weights of that path.
4) Apply Genetic Algorithm to the graph (SYTG).
5) Calculate the fitness value

   For each path cost(x) has been calculated.

   Apply the fitness function as $F(x) = x \times x$

Now calculate the probability for individual as P (i) =F(x)/ ΣF(x)

6) Generate the random numbers to calculate the new population.
7) Apply Crossover operation on the pair of chromosomes. Mate first two individuals together by applying single point crossover from 3rd bit from right.
8) Apply Mutation function by mutating every fifth bit only in case when random number generated is less than 0.4.
9) Whole process is repeated till the fitness value minimizes or maximum number of generations is reached or all the paths have been covered.
10) Best test path is generated or we can say that the test cases are optimized.
11) End.

7. Case Study

This section shows the case study done on the example Online Examination System. The possible paths generated from the above graph in (Fig. 4) are:

Table 1. Fitness of Initial Population

Path No.	Chromosome	*X*	*X*×*X*	Probability	Cumulative Probability	Associated bin
1	010110011	179	32041	0.115538	0.11553	0-0.2
2	011100001	225	50625	0.182550	0.29808	0.2-0.4
3	101000101	325	105625	0.380877	0.67897	0.4-0.7
4	010100001	161	25921	0.093469	0.77243	0.7-0.8
5	010100010	162	26244	0.094634	0.86707	0.8-0.9
6	011000000	192	36864	0.132929	1	0.9-1
	Sum		277320			

Table 2. Selection of New Generation

Random No.	Falls into bin	Selection	Crossover	Mutation
0.67645	3	101000101	101000101	101000101
0.65438	3	101000101	101000101	101000101
0.73425	4	010100001	010100010	010100010
0.81764	5	010100010	010100001	010100001
0.14521	1	010110011	010110001	010100001
0.75671	4	010100001	010100011	010100011

Table 3. Fitness of Next Population

Path No.	Chromosome	*X*	*X*×*X*	Probability	Cumulative Probability	Associated bin
1	101000101	325	105625	0.334356	0.334356	0-0.2
2	101000101	325	105625	0.334356	0.668712	0.2-0.5
3	010100010	162	26244	0.083075	0.751787	0.5-0.6
4	010100001	161	25921	0.082053	0.833840	0.6-0.8
5	010100001	161	25921	0.082053	0.915893	0.8-0.9
6	010100011	163	26569	0.084104	1	0.9-1
	Sum		315905			

Table 4. Selection of New Generation

Random No.	Falls into bin	Selection	Crossover	Mutation
0.17854	1	010110011	010110001	010100001
0.74135	4	010100001	010100011	010100011
0.61875	3	101000101	101000101	101000101
0.60914	3	101000101	101000101	101000101
0.82129	5	010100010	010100011	010100011
0.61225	3	010110011	010110010	010110010

Table 5. Fitness of Next Population

Path No.	Chromosome	X	$X \times X$	Probability	Cumulative Probability	Associated bin
1	010100001	161	25921	0.080677	0.080677	0-0.2
2	010100011	162	26244	0.081669	0.162346	0.2-0.4
3	101000101	325	105625	0.328698	0.491044	0.4-0.6
4	101000101	325	105625	0.328698	0.819742	0.6-0.8
5	010100011	162	26244	0.081669	0.901411	0.8-0.9
6	010110010	178	31684	0.098598	1	0.9-1
	Sum		321343			

From these tables it may be observed that, the fitness value has been changed in two generations. The difference between the values of the chromosomes between two generations also started decreasing. We noticed that the new population again contains the test case 3. By further calculations; we observed that the test case 3 is the optimized test case.

8. Conclusion and Future Work

UML diagrams are converted into graphs and then are integrated to generate the test cases. Genetic algorithm is also applied to optimize the test cases. The proposed model covers maximum number of faults. So, time and cost for test case generation is also minimized. The approach is applied on Online Examination System. In our future aspects the proposed approach can be automated. Also the same approach can be applied on different UML models.

References

[1] Free Uml Diagram Tool. Retrieved, from http://www.uml-diagrams.org/use-case-diagrams.html

[2] Mall, R. (2009). *Fundamentals of Software Engineering* (3rd edition.). Prentice Hall.

[3] Tripathy, A. M. (2013). Test case generation using activity diagram and sequence diagram. *Proceedings of ICAdC* (pp. 121– 129).

[4] Monalisa, S., Debashish, K., & Rajib, M. (2007). Automatic test case generation from UML sequence digrams. *Proceedings of IEEE Conference on Software Maintenance*.

[5] Namita, K., & Chhillar, R. S. (2015). Test case generation and optimization using UML models and geneic algorithm.

[6] Ajay, K. J., Santosh, K. S., & Durga, P. M. (2014). A novel approach for test case generation from UML ativity diagram. *Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques*.

[7] Ranjita, K. S., Vikas, P., & Prafulla, K. B. (2013). Generation of test cases using activity diagram. *Interntional Journal of Computer Science and Informatics*, *3(2)*.

[8] Namita, K., & Chhillar, R. S. (2014). Literature review of test case generation technique for object orented system.

[9] Swagatika, D. (2012). Test case generation for concurrent object oriented systems using combinational UML models. *Arup Acharya and Durga Prasad Mohapatra*.

Namita Khurana has done her graduation from the Kurukshetra University, Kurukshetra, Haryana. She received her M.Sc(I.T) from Guru Jambeshwar University Hisar, Haryana. She received her M.Phil from C.D.L.U, Sirsa, Haryana. Now she is working as a Ph.D research scholar in Maharishi Dayanand University, Rohtak, Haryana, India. Her research interests include software engineering, soft computing and artificial intelligence. She has a total of more than six years of teaching experience.

Rajender Singh Chhillar is a professor at the Department of Computer Science at Maharshi Dayanand University, Rohtak, Haryana, India. During his service in Maharshi Dayanand University, Rohtak. He served as the director at the University Institute of Engineering and Technology; a director at the Computer Centre; He served as the head at the Department of Computer Science, Chairman, a board of studies; a member, a executive and a academic councils. His research interests include software engineering, software testing, software metrics, web metrics, bio metrics, data warehouse and data mining, computer networking, and software design. He has published more than 91 journal and 65 conference papers over the last several years and has also written two books in the fields of Software Engineering and Information Technology. He has visited many countries including France, Hong Kong, China, U.K, Dubai and Nepal. He also won the best paper award in International Conference ICCEE- 2013 held in Paris, France during October 12-13, 2013.

Professor Chhillar is a director of board, CMAI asia association, New Delhi and a senior member of IACSIT, Singapore and a member of Computer Society of India. Professor Chhillar has been serving as a editorial board member, guest editor and reviewer of multiple international journals, and serving as a program committee chair, keynote speaker and session chair of multiple international conferences. He also performs advisory work to Government agencies and academic bodies.

Usha Chhillar is working as a head at the Department of Computer Science, A.I.J.H.M. PG College, Rohtak, Haryana, India. She obtained her Ph.D degree in computer science from the Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. She pursued her master degree in computer science from Maharshi Dayanand University (MDU), Rohtak and the M.Phil (computer science) from Ch. Devi Lal University (CDLU), Sirsa. She has total more than thirteen years teaching experience. Her research interests include software engineering, object-oriented and component- based software metrics.