An Efficient Algorithm for Evaluation of Object-Oriented Models

Manoj Kumar
Sr. Asst. Prof.
Dept. of IT,
IISE, Lucknow, INDIA

Dr.Mohammad Husain
Professor, AIET,
Lucknow, INDIA

Gyanendra K. Gupta
Asst. Prof., KIT,
Kanpur, INDIA

Amarjeet Singh
Asst., Prof., IEM
Lucknow, INDIA

ABSTRACT
In the current scenario of software development, the object-oriented technology has become the de-facto for development. It has become very popular and has been proved to be highly useful in software development process.

Genetic algorithm is a very effective optimization tool for many engineering application problems. There are many applications existing in genetic algorithm, but no one is with object-oriented systems. This paper approaches the application of genetic algorithm in object-oriented models. When we implement any application of genetic algorithm with object-oriented system, it increased the efficiency of system and gain momentum due to the availability of worthless processing power in any application. Object-oriented design, give a more natural representation of any kind of information. It has one more advantage of better memory management and code reusability. It would be very useful to work on defining methods to evaluate different object oriented models. To achieve the same, we proposed a genetic algorithm to evaluate object-oriented model.

Keywords
Genetic algorithm, object diagram, binary tree, crossover, object-oriented.

1. INTRODUCTION
Software modeling approach is a very important part of software development. Software industry expanded many times for this process. Well designed software can solve the very complex problem in a better way. Object-oriented model has become the standard analysis and design phases within a software development process, where object-oriented modeling approaches are becoming more and more the standard ones.

UML plays a very common role in the object-oriented modeling [1]. Object-oriented modeling approach around the whole software engineering life cycle, the well-known software engineering principles and qualities explained in [2], certainly apply. In the particular context of object-orientation, however, we can be substantially more specific about many of these principles and qualities. So, we start studying object-oriented modeling by re-opening the discussion on software engineering principles and qualities. Object oriented modeling to be sufficiently user-friendly to all kinds of possible standards. That is, for all clients of any model, its relevant parts expressed in the modeling language, must be understandable, and must be clear even. For the modeler as well as for all other persons involved in the modeling activity, any model must be expressive, precise and clear as well.

In particular, the general software engineering principle of separation of concerns combined with object-oriented modeling characteristics has turned out to be very useful. The basic idea of object-orientation is the consequent application of the abstract data type concept, combining data and functionality. Secondly, it advocates their integration. Object-oriented modeling gives a particular meaning to the software engineering principles of modularization and of separation of concerns. It should lead to full scalability of the modeling, since a whole model can be considered as one package, with the properties of a class, which can be integrated in a larger model in a consistent manner.

Object-oriented model is using in all areas and it is dominated by the Unified Modeling Language (UML)[3]. Finally, this language was accepted as industrial standard in November, 1997 by the OMG (Object Management Group). UML was developed as solution to the so-called object-oriented method war, which grow up in the beginning of the 1990s, where more than fifty different-different object-oriented modeling approaches could be identified in the software industry.

Object-oriented design is a design strategy, where system designers think in terms of 'things' instead of operations or functions. This executing system is made up of interacting objects that maintain their own local state and provide operations on that state (Figure 1)[4].

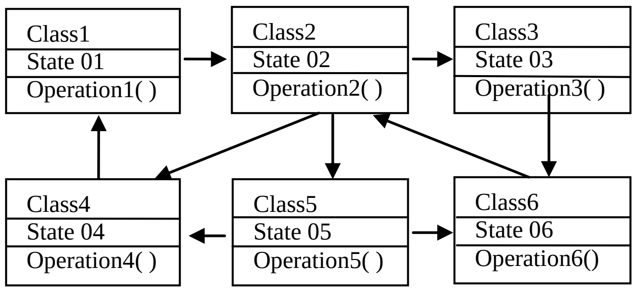


Figure 1: System Made Up of Interacting Objects

The Genetic algorithm is an adaptive heuristic search method based on population genetics. Genetic algorithm was introduced by John Holland in the early 1970s [5]. This algorithm works by maintaining a population of potential solutions to the population. In each generation or iteration every potential solution is evaluated to determine, how well it for solution of problem. The best individuals are picked out and either "recombined" by swapping parts of the strings or taken with no change from the population of the next generation. The algorithm stops after a specific number of generations or when a suitable solution is found [6]. Genetic algorithm is based on probability search. It is based on the process of natural selection and natural genetics. Genetic algorithm is started with a set of solutions called population. A solution is represented by a chromosome. The population size is preserved throughout each generation. At each generation, fitness of each chromosome is evaluated, and then chromosomes for the next generation are probabilistically selected according to their fitness values. Some of the selected chromosomes randomly crosses and produce offspring. While producing offspring, crossover and mutation randomly occurs. Chromosomes with high fitness values have high probability of being selected; chromosomes of the new generation may have higher average fitness value than those of the old generation. The process of evaluation is repeated until the end condition is satisfied. The solutions in genetic algorithms are called chromosomes or strings [7]. In most cases, chromosomes are represented by lists or strings**.** A genetic algorithm is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Genetic algorithms have been used to find optimal solutions to complex problems in various domains such as biology, engineering, computer science, and social science.

In this paper, our approach uses the genetic algorithm in evaluation of object-oriented models. We use class diagrams represented in the form of a tree. In section 2, a review of genetic algorithm with object-oriented techniques is presented. In section 3, we show the proposal of new development method. In Section 4, we give an object diagram for a case study. Finally, section 5 presents conclusion and future work.

2. RELATED WORK

Chartchai Doungsa-ard et al[8] have proposed to generate test data from UML state diagram, so that test data can be generated before coding. He implemented to generate sequences of triggers for UML state diagram as test cases using genetic algorithm. His proposed algorithm has been demonstrated manually for an example of a vending machine. C. S. Krishnamoorthy et al [9] discusses the object-oriented design aside from give a more natural representation of information, he also facilitates better memory management and code reusability and his team shows how classes derived from the implemented libraries can be used for the practical size optimization of large space trusses, where several constructability aspects have been incorporated to simulate real-world design constraints. Strategies are discussed to model the chromosome and to code genetic operators to handle such constraints. Strategies are also suggested for member grouping for reducing the problem size and implementing move-limit concepts for reducing the search space adaptively in a phased manner. The implemented libraries are tested on a number of large previously fabricated space trusses, and the results are compared with previously reported values. Federico M. Stefanini and Alessandro Camussi[10] approach becomes feasible performing a Monte Carlo simulation of the natural evolution process, in which population improvement (search for solutions) in a considered environment (the spec problem domain) is achieved by following the genetic paradigm. Starting with a randomly constituted sample of individuals, drawn from the population of admissible values and expressed as binary strings, random mating brings about individuals of the next generation. Parents are chosen with a greater probability as the number of constraints violated by each individual becomes smaller.

To generate the UML state diagrams there is automatic test case, which has been suggested by Samuel et al [11]. All the steps associated with diagrams covered by this test case. Simple predicates can reduced the number of test cases. They have taken the example of an ice cream vending machine. But Samuel et al were not succeeded to achieve globally optimal solution using alternating variable method. So they proposed genetic algorithm to achieve the UML state diagrams.

M. Prasanna and K.R. Chandran[12] suggests a model based approach in dealing with object behavioral aspect of the system and deriving test cases based on the tree structure coupled with genetic algorithm. Genetic Algorithm's tree crossover has been proposed to bring out all possible test cases of a given object diagram.

3. PROPOSAL OF NEW DEVELOPED METHOD

We create a class diagram of ATM system by using StarUML software, showing a short picture of the elaborate condition of the system. There is tree in this system having root node and many child nodes. There is an application of genetic algorithms crossover operators producing new generation of tree. Further trees are converted into binary trees. Following steps are used in the method:

a) Making of class diagram by using StarUML software and store with .uml as extension.

b) Make a tree using class names and application of genetic algorithms crossover techniques.

c) New generation of trees are forming and converting into binary trees.

d) Passing new generation of binary tree by the use of depth first search techniques.

Illustration of the above given steps in the form of flow charts in figure 1.

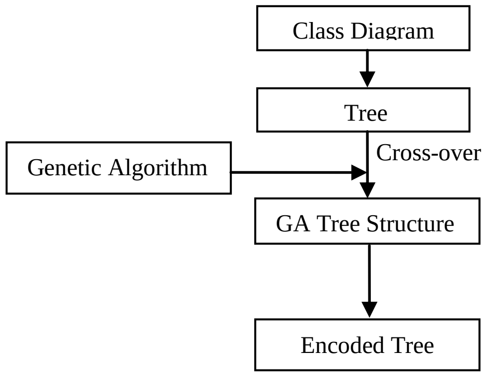


Figure 2: Flowchart of Proposed Methodology

4. CASE STUDY

A Bank wishes to introduce ATM service to provide limited facilities to his customers. Customers may get ATM cards on request. Users may view their balance or transfer or withdraw money using these cards. Cards may be used to access many accounts and an account may be accessed using different cards. A card may be blocked temporarily or permanently (e.g. If it is lost) by the Bank. A PIN is associated with each card to verify the authority of the user. A bank maintains two kinds of account for customers, one called as saving account and the other as current account. The saving account provides compound interest & withdrawal facility, but no cheque book facility .The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

4.1 Step 1 Object Diagram

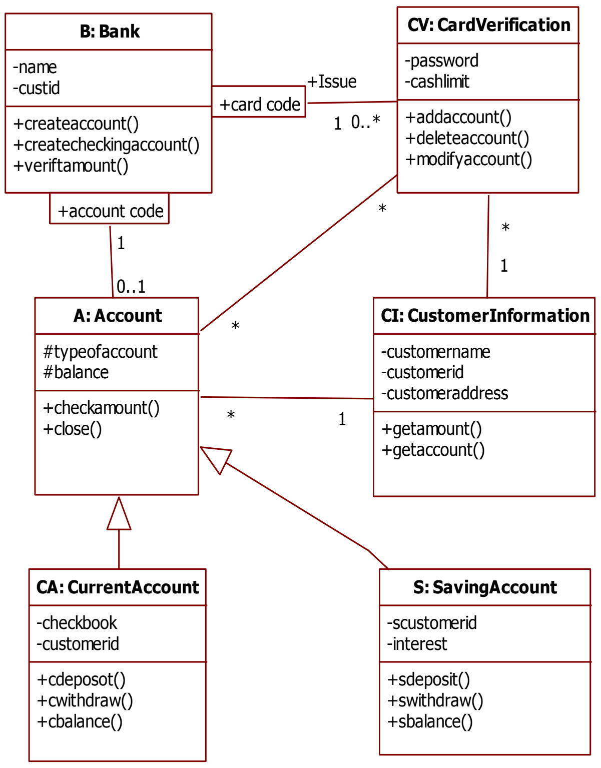


Figure 3: Class Diagram of ATM System

4.2 Step 2 Structure

Structure of tree is very much simple, easily understandable and can be stored in the computer memory. Here, we use an ATM system in a tree form. To do this we have to make the following modifications as given in figure 4 (a).

i) Nodes are showing as class and placed vertically one after the other.

ii) Attributes are arranged in left side of the branch of the corresponding node.

iii) Right side of the corresponding node taken by methods.

iv) If there is any possibility of duplication, class name is added as prefix for the nodes.

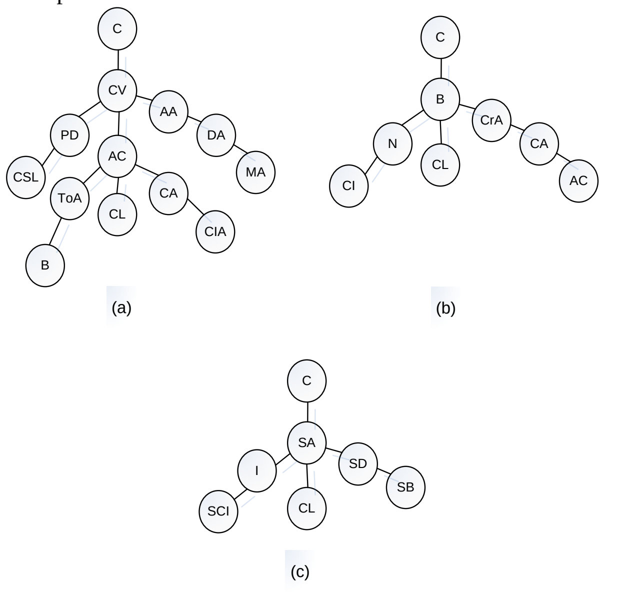


Figure 4: Converting Class Diagram to Tree Structure

4.3 Step 3
Tree Crossover

Figure no. 4(a) and 4(c) showing a crossover method which is applied on the trees. In this method first of all, one point is selected among the main trees and after crossing over it producing new generation of offspring tree.

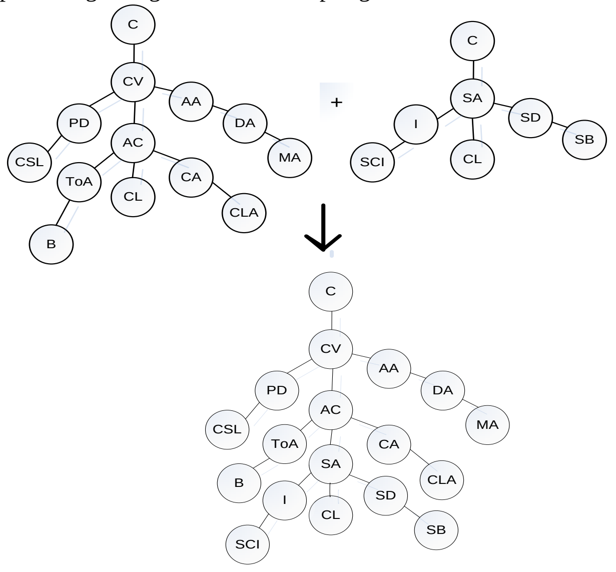
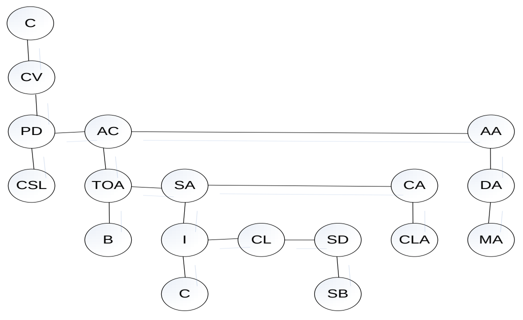


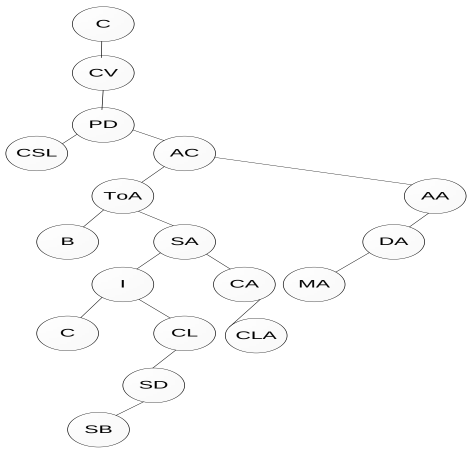
Figure 5: Tree Crossover of figure 3a & 3c in ATM System

4.4 Step 4 Converting Binary Tree

Figure 5 is not a binary tree. To make the binary tree the nodes which are placed in left branch of the root node should be vertically and its sibling in horizontal order. Now Figure 6, this way a binary tree is formed.



(a)



(b)

Figure 6: Binary Tree Form of ATM by Crossing Figure 3a & 3c

Table 1: Chromosome Mapping Information

S.No.	**Nodes**	**Object in Sequence model**
1	C	Customer
2	CV	Card Verification
3	CL	Close
4	AC	Account
5	PD	Password
6	CSL	CashLimit
7	AA	Add Account
8	DA	Delete Account
9	MA	Modify Account
10	ToA	Type of Account
11	B	Balance
12	CA	Check Amount
13	CLA	Close Account
14	BK	Bank
15	N	Name
16	CP	Check Password
17	CI	Customer Id
18	CRA	Create Account
19	CV	Card Verification
20	CIN	Customer Information
21	SA	Saving Account
22	I	Interest
23	CB	Check Book
24	SD	Saving Deposit
25	SW	Saving Withdraw
26	SB	Saving Balance
27	SCI	Saving Customer Id

5. CONCLUSION & FUTURE SCOPE

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Here, we focus on genetic algorithm in evaluation of object-oriented model. It solves the problem of optimization and increases the efficiency of a system. By this model, we also facilitate better memory management and code reusability. In future, it may carry out towards the development of UML using genetic algorithms.

6. REFERENCES

[1] G. Engels, L. Groenewegen, "Object – Oriented Modeling: A Roadmap", In Proceedings of ICSE - Future of SE Track, 2000, pp.103-116 .

[2] C. Ghezzi, M. Jazayeri, D. Mandrioli, "Fundamentals of Software Engineering", Prentice - Hall International, 1991.

[3] G. Booch, J. Zumbaugh, I. Jacobson, "The Unified Modeling Language User Guide", Addison - Wesley, Reading, MA., 1999.

[4] I. Sommerville, "Software Engineering", 6th Edition, Chapter 12, 2000.

[5] L. Tsoukalas, and R. Uhrig, "Fuzzy and Neural Approaches in Engineering", Wiley, 1997.

[6] J. R. Koza, M. A. Keare, M. J. Streeter, W. Mydlowec, J. Yu, G. Canza, "Genetic Programming IV: Routine Human Competitive Machine Intelligence", Kluwer Academic Publishers, Norwell, MA, 2003.

[7] W. Lee, H. – Yung Kim, "Genetic Algorithm Implementation in Python", Electronics and Telecommunications Research Institute, ACIS International Conference on Computer and Information Science, IEEE, 2005, pp. 8 - 11.

[8] C. Doungsa - ard, K. Dahal, A. Hossain, and T. Suwannasart, "An Automatic Test Data Generation from UML State Diagram Using Genetic Algorithm", Proceedings of International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2006, pp. 1-5.

[9] C. S. Krishnamoorthy, P. P. Venkatesh, and R. Sudarshan, "Object - Oriented Framework for Genetic Algorithms with Application to Space Truss Optimization", Journal Computer in Civil Engineering, vol 16, no 1, 2002, pp. 66-75.

[10] F. M. Stefanini and A. Camussi, "APLOGEN: An Object Oriented Genetic Algorithm Performing Monte Carlo Optimization", Oxford Journal , Life Science Bioinformatics, vol 09, no 06, 1993, pp 695- 700.

[11] P. Samuel, R. Mall, and A. K. Bothra , "Automatic Test Case Generation Using UML State Diagrams", IET Software, 2008, pp. 79-93.

[12] M. Prasanna1 and K. R. Chandran, "Automatic Test Case Generation for UML Object diagrams Using Genetic Algorithm", International Journal Advance Soft Computing Application, vol 1, no 1, 2009, pp. 19-31.