Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Optimized Based Framework for Analysis and Designing

Manoj Kumar*
Department of IT, IISE,
Lucknow, India

Prof. (Dr.) Mohammad Husain
Jahangirabad Institute of Technology,
Barabanki, India

Abstract*— Description of an object-oriented based framework aimed at the development, experimentation, and studies of accuracy and optimization problems. To analyze the problem that is two views. In the first view, we evaluate the object-oriented model using the genetic algorithm and in the second view, analyzing the design for any errors by mutation operator and finally the fault free design is obtained.

Keywords*— Genetic algorithm, Object-oriented, Mutation analysis, UML-HAZOP, Multi-view

I. INTRODUCTION

Well designed software can solve the complex problem in a simple way. Object-oriented model has become the standard analysis and design phases within a software development process, where object-oriented modelling approaches are becoming more and more the standard ones. Presently, developing software is an up- growing business globally. Often, the development of object oriented models using UML notation is attaining more popularity in the market of software engineering. But along this, one major problem is presence of errors that can cause some trouble to the user in future. We can overcome from these problems, but one truth is also seen that is recognition of errors and their correction demanding enough time and money. Nowadays many companies hiring such professionals, those, who are experts in troubleshooting and they have paid high for this only. If these defects are not found out at the right phase of development and delivered to the client with some defects, resulting the huge loss to the company 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. In this paper, very effective optimization tool is used and solved many engineering application problems.

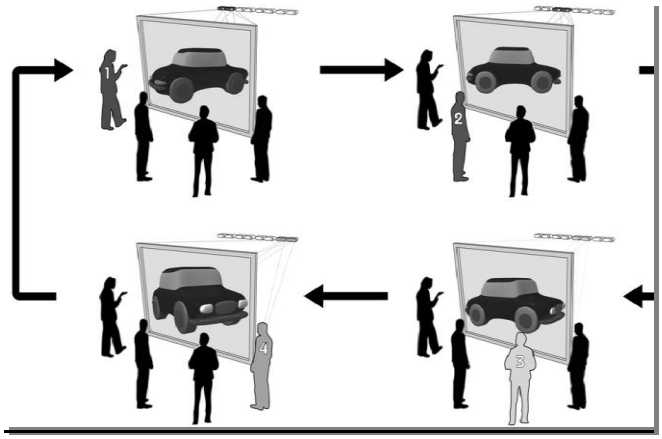

Fig 1: Multi-view Angle of One Picture

Fig 1 has shown the many different angles, while buying a car. One question may arise of seeing a car in a multi-view to proof these concepts, an example of car buying a car taken. To get more about different feature of car, it is important to see a car from multiple angels. Same here in this paper, the object-oriented model is seeing by multiple views concepts. Very minute fault can be seen by taking the help of multi-view. To make the very clear picture of object-oriented model by multi-view, the genetic algorithm and UML-HAZOP technology [1] is used here.

The Genetic algorithm is an adaptive heuristic search method based on population genetics. Genetic algorithm was introduced by John Holland in the early 1970s [2]. 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 [3]. Genetic algorithm is based on probability search. It is based on the process of natural selection and natural genetics.

.II. Related Work

J. R. R. A. Martins et al [4] present MDO, an object-oriented framework that facilitates the use of algorithms for multidisciplinary optimization (MDO). The Optimization class represents a single optimization problem. Chartchai Doungsa-ard et al [5] 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.

C. S. Krishnamoorthy et al [6] 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.

Parvinder S. Sandhu et al [7] evaluates performance of genetic algorithm based classification technique in predicting fault prone classes using open source software. The proposed GA based classification technique shows 80.14 percent accuracy.

Marion R. Finley et. al [8] have developed a distributed object-based modeling environment C‘DOME” to aid designers involved in distributed design modeling. He has ([9], [10], [11]) developed an approach for unit and integration testing, which focuses on testing the APIs of software components that form part of the server-side application or business logic.

Sunwoo Kim et al. [12] propose the use of a safety technique known as HAZOP (Hazard and Operability Studies) to rigorously generate mutation operators for Java. A set of Java mutation operators is proposed by applying HAZOP to the Java syntax definition and is compared to the operator sets of current mutation systems. Janusz GÓRSKI et al. [1] present a method supporting detection of defects in UML based software documentation. This method named UML-HAZOP is the adoption of HAZOP (Hazard and Operability Studies) – a technique widely applied to safety-related systems, and concentrates on analyzing “flows” between system’s components in order to detect anomalies related to these flows.

Janusz Górski, Aleksander Jarzębowicz[13] introduces the UML-HAZOP and presents results of its validation through a series of case studies and controlled experiments.

They ([14], [15]) are interested in using mutation to evaluate, compare, and improve quality assurance techniques for concurrent Java. The use of mutation with Java has been proposed in previous work – for instance the MuJava tool. MuJava includes two general types of mutation operators for Java: method level operators [14] and class level operators [16]. A pilot study that used a pre-existent mutant operator set confirmed the need of mutant operators specifically designed for the Java model of concurrency and synchronization [17].

M. Prasanna and K.R. Chandran[18] suggests a model based approach in dealing with object behavioural aspect of the system and deriving test cases based on the tree structure coupled with genetic algorithm.

III. UML- HAZOP METHOD

A Hazard and Operability (HAZOP) study is a structured and systematic examination of a planned or existing process or operation in order to identify and evaluate problems that may represent risks to prevent efficient operation. The HAZOP technique was initially developed to analyze chemical process systems, but has later been extended to other types of systems and also to complex operations and to software systems. A HAZOP is a qualitative technique based on guide-words and is carried out by a multi-disciplinary team (HAZOP team) during a set of meetings.

The HAZOP study should preferably be carried out as early in the design phase as possible - to have influence on the design. On the other hand; to carry out a HAZOP rather complete design is required [19]. As a compromise, the HAZOP is usually carried out as a final check when the detail design has been completed. A HAZOP study may also be conducted on an existing facility to identify modifications that should be implemented to reduce risk and operability problems.

A. UML-HAZOP Inspection

HAZOP (Hazard and Operability Study) is widely used in the safety-critical systems domain to seek for possible hazards, the events or states that, if occur, can have dangerous consequences. The technique is focusing on system models and analyses them by a systematic application of HAZOP guidewords to the model components to identify potentially dangerous situations in the system. HAZOP was first introduced in chemical industry and later was adapted for software safety [20,21]. An example set of HAZOP guidewords is given in table 1 Application of the guidewords to the elements of the analyzed model results in suggestions of potential hazards, for instance the guideword MORE applied to the temperature of a pipe results in “too high temperature in the pipe” and applying LATE to a transition between states suggests “the transition is fired later than expected”. Such suggestions can then be analyzed, if they constitute system hazards. An important observation, which can be made with respect to HAZOP is that it comprises two phases that can be considered in separation. The first phase is focused on detecting anomalies, i.e. the possible (but unwanted) states or events that can occur in a considered system. The second phase is concerned the assessment of those anomalies and is driven by a set of chosen criteria.

Table 1: Generic HAZOP Guidewords

Guideword	Generic Interpretation
NO	The complete negation of the design intention. No part of the intention is achieved and nothing else happens.
MORE	A quantitative increase.

LESS	A quantitative decrease.
AS WELL AS	All the design intention is achieved together with additions.
PART OF	Only some of the design intention is achieved.
REVERSE	The logical opposite of the intention is achieved.
OTHER THAN	Complete substitution, where no part of the original intention is achieved but something quite different happens.
EARLY	Something happens earlier than expected relative to clock time.
LATE	Something happens later than expected relative to clock time.
BEFORE	Something happens before it is expected, relating to order or sequence.
AFTER	Something happens after it is expected, relating to order or sequence.

IV. OUR METHODOLOGY

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

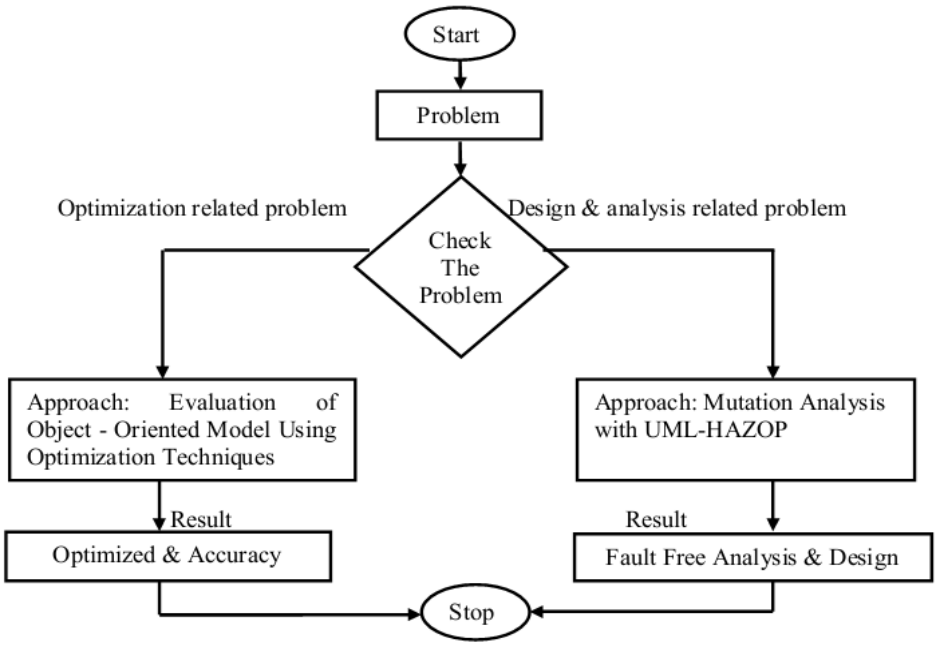

Fig 2: Problem and Solution in Flow Chart Form

Here two approaches are used to solve the problem. Problem is mentioned in fig 2 and checked the feasibility of the solution by using different techniques and result the outcome. The optimization technique genetic algorithm is used to evaluate the object-oriented model [22]. This genetic algorithm worked 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. It 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 [23]. 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. It is categorized as global search heuristics.

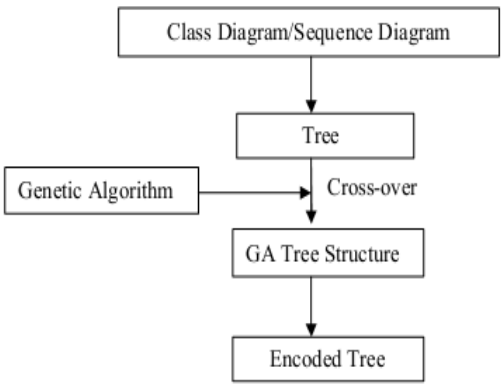

Fig 3: Flowchart of Proposed Methodology

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 (fig 3). 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.

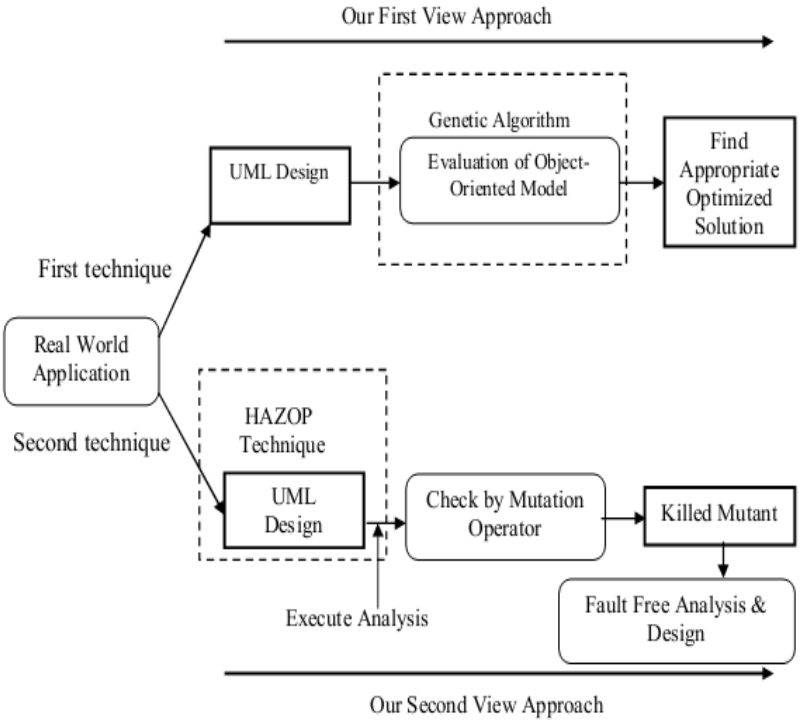

Fig 4: Multi-view System Approach

Fig 4 has shown the approach using multi-view technique. A real world application is taken in both views and solved by two different techniques. The above fig has shown the first view, evaluating the object-oriented model using the genetic algorithm. UML design utilizing UML-HAZOP technique is shown in the second view and analyzes the design for any errors by mutation operator [24] and finally the fault free design is obtained. 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. The development of framework for safety critical area what happens, when some part of a system deviates from the intentions of designer is a critical paper issue. When HAZOP technique is applied by using UML, then, the object-oriented design is checked with a fault-free analysis and design. By mutation analysis and HAZOP, a better optimum result. The mutation method is a fault-based testing strategy that measures the quality/adequacy of testing by examining whether the test set (test input data) used in testing can reveal certain types of faults. This paper described the UML-HAZOP technique with mutation based operator or

analysis. The aim of this paper is to noticing the defects that comes during the early phase of software development and making sure that the fault free software is delivered to the user. The purpose of choosing this topic like defects introduced in the object oriented models expressing UML notation, because it is triggered/interpreted by man itself. In this paper, the framework for performing mutation analysis and deviants with UML-HAZOP technique approach is used for evaluating the object-oriented model (fig 4).

V. Conclusion

The final goal of this paper is to develop a framework for fault free analysis and find an exact or optimum result for the optimization. The general objectives are:

1. The ultimate target of the paper is to find an approach to the application of the optimization technique with UML. It increased the efficiency and effectiveness of system and gain momentum due to the availability of worthless processing power in any application. It is a technique used in computing to find exact or approximate solutions to optimization problems.
2. To develop a framework for safety critical area, if some part of a system deviates from the intentions of designer is a critical paper issue. A framework to check the object-oriented design with a fault-free analysis is the beneficiary result of the paper. This paper describes a technique with mutation based operator or analysis. Using this methodology much optimum result and solution is found. At the end of the paper a better analysis and design for real world application is obtained.

REFERENCES

[1] J. Górski, A. Jarzębowicz, *Detecting Defects in Object – Oriented Diagrams Using UML - HAZOP*, Foundations of Computing and Decision Sciences, Vol. 27, No. 4, 2002.

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

[3] 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.

[4] J. R. R. A. Martins, C. Marriage, 3rd AIAA *Multidisciplinary Design Optimization Specialist Conference*, University of Toronto Institute for Aerospace Studies, Toronto, April 23– 26, 2007.

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

[6] 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, pp. 66-75, 2002.

[7] P. S. Sandhu, S. K. Dhiman, A. Goyal , *A Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes*, World Academy of Science, Engineering and Technology, pp. 60, 2009.

[8] M. R. Finley, H. Akimaru, K. Yaniori, *On the Design of Tele-Learning Systems Using Genetic Algorithms*, Communication Technology Proceedings, WCC - ICCT 2000.

[9] J. Hartmann, C. Imoberdorf, and M. Meisinger, *UML-Based Integration Testing*, Proceedings of ISSTA 2000, pp. 60-70, 2000.

[10] J. Hartmann, M. Vieira, H. Foster, A. Ruder, *UML-based Test Generation and Execution*, Siemens Corporate Research, 2004.

[11] M. Meisinger, *Automatic Test Case Generation From Communicating State Machines*, M.S. Thesis, Technical University Munich, 2000.

[12] S. Kim, J. A. Clark, and J. A. McDermid, *The Rigorous Generation of Java Mutation Operators Using HAZOP*, In Proceedings of the 12th International Conference on Software and Systems Engineering and their Applications (ICSSEA' 99), Paris, France, Dec-1999.

[13] J. Górski, A. Jarzębowicz, *Development & Validation of a HAZOP Based Inspection of UML Models*, 3rd World Congress for Software Quality, Sep. 26-30, 2005.

[14] J. O®utt, R. Alexander, Y. Wu, Q. Xiao, and C. Hutchinson, *A fault model for subtype inheritance and polymorphism*, In Proceedings of the 12th International Symposium on Software Reliability Engineering, pp. 84-93, 2001.

[15] Jeremy S. Bradbury, James R. Cordy, Juergen Dingel, *Mutation Operators for Concurrent Java* (J2SE 5.0), Canada, 2006.

[16] Gill, P. E., Murray, W., and Saunders, M. A., SNOPT: *An Sqp Algorithm For Large-Scale Constrained Optimization*, SIAM Journal of Optimization, Vol. 12, No. 4, pp. 979–1006. 2002.

[17] M. Delamaro, M. Pezz`e, A. M. R. Vincenzi, and J. C. Maldonado, *Applying Mutation Testing to Multi-threaded JAVA Programs*, Tech Report, 2001.

[18] 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.

[19] M. Rausand, *HAZOP Hazard and Operability Study*, System Reliability Theory, 2nd Edition, Wiley, 2004.

[20] UK Ministry of Defense, Defense Standard 00-58, *HAZOP Studies on Systems Containing Programmable Electronics (Part 1&2)*, No. 2, 2000.

[21] F. Redmill, M. Chudleigh, J. Catmur, *System Safety: HAZOP and Software HAZOP*, J. Wiley & Sons, 1999.

[22] M. Kumar, Dr. M. Husain, G. K. Gupta, A. Singh, *An Efficient Algorithm for Evaluation of Object-Oriented Models*, International Journal of Computer Applications (0975 – 8887) Volume 24– No.8, June 2011.

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

[24] M. Kumar, Dr. M. Husain, *A Framework for Performing Mutation Analysis and Deviants*, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1, Issue-4, September 2011.

