Automatic Test Cases Generation using Multistage-Based Genetic Algorithm for Object Oriented Testing

Anju Bala, Rajender Singh Chhillar

Abstract: ***Software testing is a major phase that takes place under the construction of software designing. Basically, testing is a process that assists in the determination of work that it reached to the desired output or not. It generally depends on the validation and verification procedure, whereas in simple terms a software testing process is to discover the bugs, errors, faults of the developed software and manage it. It is also considered as the risk based activity. The testing criterion is different at each level and it is completed in various steps. The life cycle of software testing is composed of various steps as the feasibility study, data gathering and specification, design or framework, unit testing, integration and system testing. At last the maintenance is occurring to finalize the software application. In software engineering several kinds of testing strategies are utilized as black box, white box, regression testing, static, dynamic and so on. There are enormous advantages of software testing. The common advantages are to investigate software quality, access the huge pool for verification, deducted the construction cost, improve the reusability, aimed at the basic competencies, increase the demand of the product, balance the time period for the development of software and boost the competitiveness. But there are also certain vulnerabilities related to the large investments, software tools, training, need of more manpower, most time consuming of test preparations, need of more testing space, hidden errors impact on the entire code and cost. In the proposed work, the performance is reliant on the better way. Test case generation is a procedure to generate software corresponding various test case generations and validate various test cases. So that research work identifies the quality of software. This process also declined the maintenance cost (MC) of a software system. In the proposed architecture design, Multi-stage Genetic algorithm has various benefits as it is highly effective in higher dimensional spaces, more memory efficient and versatile. Basically, Multi-stage GA is applied in several real-time applications as in the text categorization, classification of test cases and regression related issues. In the research work, mutants compare various existing techniques and performance parameters are like as mutants, accuracy rate, time consumption and number of events. The planned approach is best in terms to enhance the accuracy rate and achieved it in a reduced time period. Several techniques are used to compare the number of events fire. So that, the architecture accuracy rate has achieved this based on the number of events. The multistage GA test case is an intelligent approach and supportive to various languages like .Net, Java, C++ and Project Management used in an automatic test case. It helps to improve the quality of software and based on the mutants. Basically, mutants are like failure (Some time it is passed or sometimes it fails). The reduced number of mutants increased the software quality.***

Index Terms: ***SDLC (Software Development Life Cycle), OOPS (Object Oriented Programming System), GA (Genetic Algorithm), PSO (Particle Swarm Optimization), ACO (Ant Colony Optimization), BCO (Bee Colony Optimization.***

I. INTRODUCTION

Testing has been acknowledged as a vital section of the (SDP) software development process. ST (Software testing) is a significant SQA(software quality assurance) event to confirm that the advantages of OOP(object oriented programming) will be appreciated [1]. Testing is a procedure that checks out the correctness and dynamic behavior of a program at the time of its execution. The main purpose of testing to ensure the program meets its desired requirements [2]. It is a most beneficial phases in SDP(Software Development Process). Its make certain that the implemented software fulfills all user requirements and execute with-out error. Software development paradigm and methods have moved from start water-fall SD to object oriented design and some other novel concept. Software testing has also moved from CT (Conventional Testing) towards OOT (Object Orientation Testing) [3]. Software testing is basically a collaboration of three Ps which means, of the phases, processes and principles of software testing. Three Ps are described as phases of testing, the process of testing and principals of testing [4]. Software testing classified into two forms as static and dynamic testing. The static testing consists of inspection of code, program and model verification. On the other side, dynamic testing processed takes input and performs the test processes and generates the automatic testing [5]. Unit testing means test each operating as a part of a class hierarchy. It is different from conventional testing. In conventional testing, the main focus is on Input-process-output. But in class testing focuses on each method. Validation testing checked that is we building the right product? Validation becomes succeeds when software function in a manner that can be reasonably expected by the customer. Validation testing is focused on the user visible action and recognizable output. Integration testing is the testing of the interaction between object oriented components. Integration testing assumes that unit testing has been done

Revised Manuscript Received on September 25, 2019

Anju Bala, Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak; Email: anjunarwal024@gmail.com
Rajender Singh Chhillar, Department of Computer Science and Applications Maharishi Dayanand University, Rohtak; Email: Chillar02@gmail.com

on the components and that the defects have been removed. System testing concerned with the execution of test cases to evaluate the whole system with respect to the user's requirement. System test checks for unexpected interaction between the units and also evaluate the system for compliance with functional requirements [6], [7], [29], [30]. There are countless testing categories. The advantages of software testing are cabled to detect faults and errors in the software development, identify the bugs, generates highly accurate results and efficiency, it also performed root cause analysis, create defect reports and improved them, clearly identified the requirements, expelled the texture execution time and flourished the values and so on [8].

A. Materials and Methods in Software Development Life Cycle

The process of software known as software life cycle that obtained the structure of software products. The processes are same as life to software so it named life cycle. It consists of the development of software and maintenance of software products. [9]. SDLC is well organized structure and a sequence of numerous steps in the software engineering to create powerful software [10]. It involves certain steps that are data collection, feasibility study, analysis, software framework, coding, testing, implementation and maintenance. There are various software development life cycles that are discussed in the following section.

Waterfall Model: This model is a classical model used in the software development life-cycle. Waterfall model was exposed by Royce in 1970 and after that defined by Boehme in mid of 1970's. The main focus is to specify the requirements of a system before software designing, it collaborates the components to interact with each other, and it tracks the progress in the development of software [11] , [31]. A waterfall is the older model which consists of overlapping steps. It played out a basic platform for various other models.

The main characteristics of Waterfall Model: In software development life cycle, waterfall model introduced numerous characteristics such as Understanding and implementation is easier rather than other models, it is the oldest model so known as global, support to weak teams, the performance is much better on mature software products, cost-effective, consists of the basic steps which further used by other models [9] , [32].

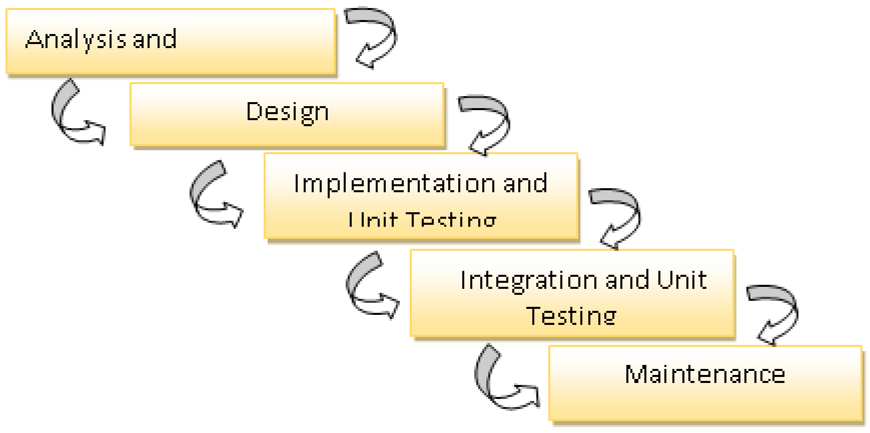

Figure 1. Classical Waterfall Model

Iterative waterfall model: Iterative model is sometimes called increment process model. It is an enhanced version of the waterfall model. In this model no confusion about the final product which can be delivered in phases as the requirements. The number of cycles is performed and the product displayed in front of the customer to get the feedback. If the developed software, selected by the customer then delivered otherwise it is modified again. The increment process model contains similar phases as the waterfall model.

The steps in the increment waterfall model are related to waterfall model. The only difference is seen in the variety of iterations. The process of developing software is associated with different the alterations. A final product generated at the end of the iteration. This is the biggest feature of the iterative model. Features of Iterative Waterfall Model are the entire projects partitioned into small parts, valuable feedbacks, feedbacks of one phase become the input to another phase, worked well even in the lack of many staff, easy implementation [9], [12], [33].

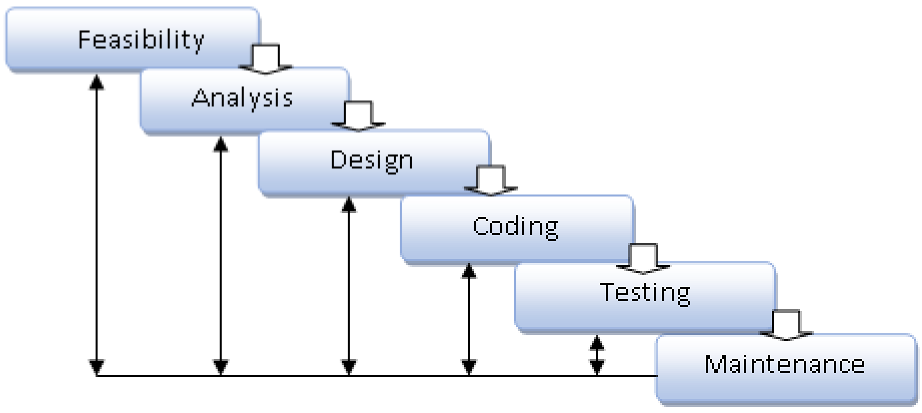

Figure 2. Iterative Waterfall Model

Prototype Model: Prototype model plays out a crucial role in SDLC and it is a model based on the extension of an incremental or iterative model. In this model, a number of prototypes inclined. The initial prototype assessed by the customers that involved well-defined documentation. Further, this prototype generates an overall architecture to software. Addition to this, the software product development through prototype model is nearest to the customer requirements [13]. Features of evolutionary or prototype model-collect user ideas and requirements, highly accurate results, feedback collection in the beginning stages, early design, coding and analysis, software success rate is extremely good.

Spiral Model: The spiral model is known from several years ago and it is an improvement and refinement of classical waterfall model mainly referred to in the huge government software. In each cycle, a prototype created to verify the needs and validation for testing. In case the risks are alleviated , then it moved to iterative waterfall model otherwise, again identified in the spiral model to alleviate the risks. Advantages of spiral model- each cycle is complete and performs better iterations rather than any other model select the mixed approaches to find out errors and risks, measurements of time and effort which assist in risk management, verification and for testing, a review of all cycles [14].

B. Soft Computing

Soft computing associated with the combination of various approaches that are capable to sort out the major issues in a computing process. These approaches are linked to fuzzy logics, probabilistic reasoning, neural networks and mainly the genetic approach. All these approaches are generating a search and find out techniques that can effortlessly deal with the complexity and real-time problems [15]. Soft computing (SC) is a critical concept which referred to a considerable speculation, mainly spotlight on the computing procedure that easily reflects the hidden details of the human brain. There are a variety of applications of soft computing. The major fields of SC are handwriting recognition, compression and image processing, automatic systems, frameworks of SC, decision support systems, Neuro fuzzy, fuzzy logic control systems, speech and visual recognition, machine learning applications, process control and many other fields [16].

Genetic Algorithm: In the mid of 1970's Holland proposed an algorithm, namely genetic algorithm that performed a random search in a particular solution space. The solutions are considered as chromosomes and determined through the fitness function [15].Operators in GA are described as below-

- Enoding: The process of representing the solution in the form of a string that conveys the necessary information
- Fitness function: A fitness function quantifies the optimality of a solution so that particular solution may be ranked against all other solution [17].
- Recombination: The process that determines which solution is to be preserved and allowed to reproduce and which ones deserve to discard.
- Cross Over: It is the process in which two chromosomes (strings) combine their genetic material (bits) to produce a new offspring which possesses both their characteristics.
- Mutation: it is the process by which a string is deliberately changed so as to maintain diversity in the population set.

Pseudo Code for Simple Genetic Algorithm

```
  Int Time
  IT := 0;
  INTIATE PPLN R (IT);   // initialize a rand
population of Chromosomes.
  Calculate fitness value;   // entire chromosomes
in a given population.
  Cal R (IT);
  Testing
  For Stop Criteria;     // it includes Time taken
and Fitness Value.

  While
  {

   Do pending operations
  }

  Increment of Time
  IT = IT + 1;

  Select
```

```
Sub PPPLN

R 0 := Choose parents G (IT);
Add genes,

Repeat process of addition

REP  G 0 (IT);
Operators       // applying certain operators.
Mutation
G 0 (IT);
Cal new Fit Value
Determine G 0 (IT);
Choose actual fit value P
Terminate GA.
```

TABLE 1: COMPARISON OF VARIOUS TECHNIQUES

Technique Used	**Description**	**Advantages**	**Applications**
Genetic algorithm	It is an algorithm which based on three basic operators, as selection, crossover and mutation search fitness value of a problem.	Solve NP hard problems Easy implementation Solve complex problems as traveling salesman problem	Health care systems Face recognition Iris recognition
Ant colony optimization	ACO approached to solve large population search space. It's based on the natural phenomena of real ants the searching for food. Ant colony optimization is dissimilar from other algorithms. It followed up to create all new set of solutions rather than modifying the existing solution. [18].	Easily balance pheromones Find out neighborhood solutions Generate reinforce and easily evaporate pheromones. Searching for neighborhood solutions.	Quardric assignments Routing of vehicles Telecommunication
Bee colony optimization	BCO is a meta heuristic technique that based on the habits of bees. For processing like real bees it introduced artificial bees as agents. The searching way of BCO is moving in the terms of iterations till reached at the desired terminal criteria. In this process each and every bee consisting of a solution for the given problem [19]	It supports to variations. Easily search optimal flights and Parallel processing, capable to search optimal flight assignment. BCO utilized to sort out the combinatorial problems.	NP Hard problems Engineering fields Find a solution of Complex lengthy problems
Particle swarm optimization	PSO is a sub-category of swarm intelligence and an evolutionary technique. PSO does not require any kind of selection operator as compared to the genetic algorithm. Entire particles are saved in the form of members of a population by run. According to researchers, the run is all about the number of evolutionary techniques that consist priority to stop [20].	Easy execution of PSO Parameters are flexible so they can fit according to the requirements. Robust process.	Max- min problems Multiobjective problems. Dynamic tracking.

II. BACKGROUND

In business and industrial works, software played out a pivotal role and is an internal area of processing. The core products of software are in banking, finance, healthcare, social networking, shopping, and e-commerce. The architecture of software product assists various things. For instance, long time period, intellectuals, domain expertise and software tools [26]. In each field, the project begins with the internal development which performed successfully to satisfy the customer needs. Sometimes the required functions don't work well that causes failures and faults in the system. It is right to say about the software failure or faults could be happening at any stage of software product development [27] and [28].

A. Software Failure

Software failure is a concept of not performing tasks properly, which generate errors in the deployment of software. It can be any kind of software product or applications that support the various applications in this technological world. In an alternative way, a software system is defined as a system of components that are interrelated. The software is a group of a variety of programs, files, documentation that are applied to make a structure of the system. The occurrence of failure arrives when the delivery services do not comply with specifications. This way of failures is seen in both hardware and software [26].

FMEA (Failure Modes and Effect Analysis): software failure mode and effect analysis is a systematic approach to identify and alleviates the failures for enhancement of reliability. Many years ago, it was used in the revealing of ranks, military, critical functions and in missile systems. The fundamental objectives of FMEA are as mentioned below:

i) Identification of single point failure
ii) Identify the redundancy area
iii) Verification of features that are required to eliminate.
iv) Identify testing for redundancy
v) For generating ranks and critical terms in the software.

Major error categories in a software system are computational, logical, data handling and interfaces [26].

B. Different Kinds of Software Failure

Software failures come under various ways which are as follow:

i) Count of Users of Applications
ii) Monetary Transactions
iii) Slow Response of Server
iv) Performance of System
v) Downloading of papers is not well[26].

C. Causes of Software Failure

In the development of software systems, there are assortments of factors that cause the failure in the software development phase. The major factors that cause failures are given below:

i. **Unrealistic Project Goals:** The system fails when there are more unrealistic and unarticulated project goals that are difficult to complete in a given time period. Due to these conditions, the processing affected and not reached the target.
ii. **Inaccurate Estimates of Resources:** Another factor that causes faults and errors in a system is seen in the access of right resources. The estimation of resources is crucial to the good deployment of a software system.
iii. **System Requirements:** The requirements are a necessity in the well-organized process of software. In case of bad requirements, the system fails.

D. Case Studies

Case Study 1: In Oxford health plans company. A failure occurred due to the newly invented billing system could not expand the business and produced faults related to un-collected payments around four hundred million from patients and one hundred and fifty million more from the caregivers.

Loss: In late 1990's the quarterly loss, prompt stock amount fell drastically from 68 dollars to 26 dollars just in one day. Afterward, the company was forced to pay 225 million dollars to settle lawsuits.

Case Study 2: Sydney Water Crop. A failure introduced in a project which referred for automatic customer information and billing, particularly for Australia's extreme water providers was abandoned in 2000. This abandon was allied to insufficient planning, a majority of change requests and cost. The total cost spent on this project was around 33 million dollars.

Case Study 3: London Stock Exchange. In 1993, efforts to design a new stock settlement system were scrapped. Later, the complexity and design along with poor management become causes of failures and it costs around 600 million dollars [28].

Case Study 4: ERP Project Failure in Jordan. The failures were generated through gaps between predictions and needs required to build on the ERP system framework along the realities of client organizations. The extremely loss of Capital and clients were un-satisfaction.

Case Study 5: This case study was related to software Ariane 5. It was a European newly rocket which was dramatically expelled after just seconds of launch in its flight. Along with this, another four cargoes of scientific satellites were also diminished. The loss was around 100 million dollars and the reputation of ESA (European space agency) was declined.

Case Study 6: Therac-25 was a Canadian cancer therapy machine developed by AECL (Atomic energy of Canada Ltd.) in the mid of 1980's. It was a software control radiation therapy to treat cancer patients. During 1985 and 1987, therac-25 give overdoses of radiations to six patients. Experts said these were given because of no dose mentioned in the system. It indecently selects the number of doses. The results show several injuries and three deaths of patients[24].

III. LITERATURE SURVEY

Jamil, M. A., [21] give a brief description of some software testing approaches. Due to the enhancement of software applications, the testing procedure was required to increment the quality of assurance. Software testing was a pivotal phase in the software development life cycle. This paper includes various terms related to software testing, which was testing methodologies, the life cycle of Software, Test optimization and quality metrics. Various techniques of software testing were introduced as follows:-

- Control flow testing
- Branch testing
- Matrix testing
- Pattern testing
- All pair testing

All these mentioned testing was deeply described and there was an enhancement of existing techniques, mainly to improve the quality assurance purposes [22]. Research on a brief description of software testing, which aimed at white box and black box testing particularly in C and C++ programs. Verification activities performed through the software development were explained as well.

From a few decades ago, the requirements of software were not much essential, but in this fast-moving world, the internet and the demand of the software increased continuously. Software development introduced for obtaining the solutions to real-life problems. For this work, several testing's and evolution was required for the software. Software testing was a process or a solution that checks out the performance of a program and system function correctly or not. Various verification activities were performed in the life cycle of software development which includes-

- Requirement Analysis
- Architecture
- Implementation
- Operations and Maintenance

In the other sections of the research paper, there were C and C++ programs with the use of the white box, black box testing was described. The traditional method applied to catch the faults in the program was faced with some limitations. To overcome these problems, white and black box testing techniques were used which was fully automatic. White box testing was called static testing that applied to software, whereas black box testing considered as dynamic testing that checked the errors on the basis of input and output. From the researched survey, it was clear that white box was performed with the conjunction of black box testing [23]. initiated research on information visualization and sequencing constraints with low-level interactions. Black box testing was in use for visualization. The main motive of this research was to develop the test approaches for visualizing and generate the rigorous evaluation of visualization [35].

Verification and verification were a procedure which was applied in the software development to check out the specifications and desired purpose fulfilled or not. This procedure includes the selection of elements, function input domain, the execution of the program. Finally, it compares the obtained output with the target output. This verification and validation procedure called software testing as well. There were two kinds of testing.

- White Box Testing
- Black Box Testing

This method was fully dependent upon constraints on the low-level interactions, which were used in visualization. The results give information about the testing of visualization requirements fulfilled with the use of black box testing. After that runtime verification was also checked throughout the black box testing [24]. Proposed a comparison between black box and white box test prioritization. Regression testing, white box and black box testing were main key terms. An experiment was done on various test prioritization techniques. At the end of this paper, there was an evidence of black and white box prioritization which was reliable and robust.

Regression testing suites were essential in terms of testing mainly for the effectiveness and robustness of faults or errors in the testing procedure. Several kinds of the testing had been proposed in this paper. However white box prioritization was well described, but black box testing wasn't compared with any other testing approach. It was applied for generating diversity in the testing techniques. Due to the less information about the black box test prioritization, it was hard to estimate its working procedure.

The experiments were examined the extreme overlap between the errors or faults found in the white box and black box test prioritization. More than that, the comparison between the performances of various testing techniques was evaluated. From the existing work, the initial testing was reliable at multiple releases. However, these new findings of black and white box testing where must give positive results in regression testing [25]. researched on the model-based method in software testing. To make the productivity and quality assurance more desirable, it required to detect faults in the initial stages of architectural design in the system. These entire requirements were necessary for the streams that need to escalate reliability of automotive companies, fabrication, Embedded software testing [34].

A model-based approach implemented to enhance the software testing. Other testing approaches were added to it for more enhancements. For instance:

- Unified Modeling Language
- Integration Testing
- Power Window Switch Module
- Approval Testing
- Hardware Testing.

All these entire approaches are working together in the model-based approach. All the specific languages were used as the input data, it evaluates the required values. Next, to it, unit cases were collaborated along with integration testing that works towards the bottom-up method and further altered it to the hardware testing.The outcome generated through model-based approach was just demanded few resources for originating software to hardware.

TABLE 2. LITERATURE SURVEY OF PREVIOUS TECHNIQUES

Author's name	**Year**	**Technique Used**	**Parameters**	**Issues**
Jamil, M. A., et al.,	2016	SDLC (Software development life cycle), automation testing, quality metrics and test driven,	-	Cost and time consumption
Nouman, M., et al.,	2016	Black box and white box testing	-	Safety critical domains
Larrea, M. L.	2017	Black box technique, user action notation and sequencing constraints.	Details on demand Test cases	Visualization implementation
Henard, C., et al.,	2016	Test prioritization of black and white and regression testing	Fault detection rate, faults, test suites, prioritization	The comparison was not earlier done.
Shin, K. W., et al.,	2018	Model based approach with X query and hardware mapping	Automatic testing	Dependency on sources.

IV. Design A Multi-Stage Genetic Algorithm (MS-GA)

The description of the research and the results are as follows-

- *Load all classes:* after start users need to upload their codes on the server to generate test cases. Here the system loads all the libraries and classes developed by the developers in the memory and transfers control to the next step for further processing.
- *Find Methods and arguments:* in the whole processing architecture, the overall processing depends upon the classes and their methods written by the developers. Due to various modules and sub-modules in the software systems, a team working together to produce high-quality software systems. This process makes various issues due to the skill and experience differences. The test case generation process validates the deployment process and provides a high-quality software system. Finding methods and argument process is used to extract various code modules from log files to generate various tests and validations.
- *Test cases:* Test case generation process used to validate the modules and find the mutants from while integration process. The developers develop codes and load all code files on the common FTP and then integrate with the existing system. Before it will integrate it needs to validate with a different branch or with other sub-module. Various test cases are used to check the different conditions and various rules satisfactions according to the application and OOPs criteria. Various mutants found by the system will issue as tickets to resolve from the back-end.
- *Processed:* due to various sub-modules in every software system, the test case generation process uses some iterative processes to process all the modules and sub-modules. This process handles module by module to form tree architecture for all the functions and sub-functions to process them in the right manner.
- *Optimization process:* the optimization process in this phase plays an important role to process the modules and generate test cases. Here the process generates various possible solutions for a module and executes them on the given code architecture. While execution, it optimizes the values of various input-output arguments according to the OOPS rules to generate the test cases and mutant from the large software system.

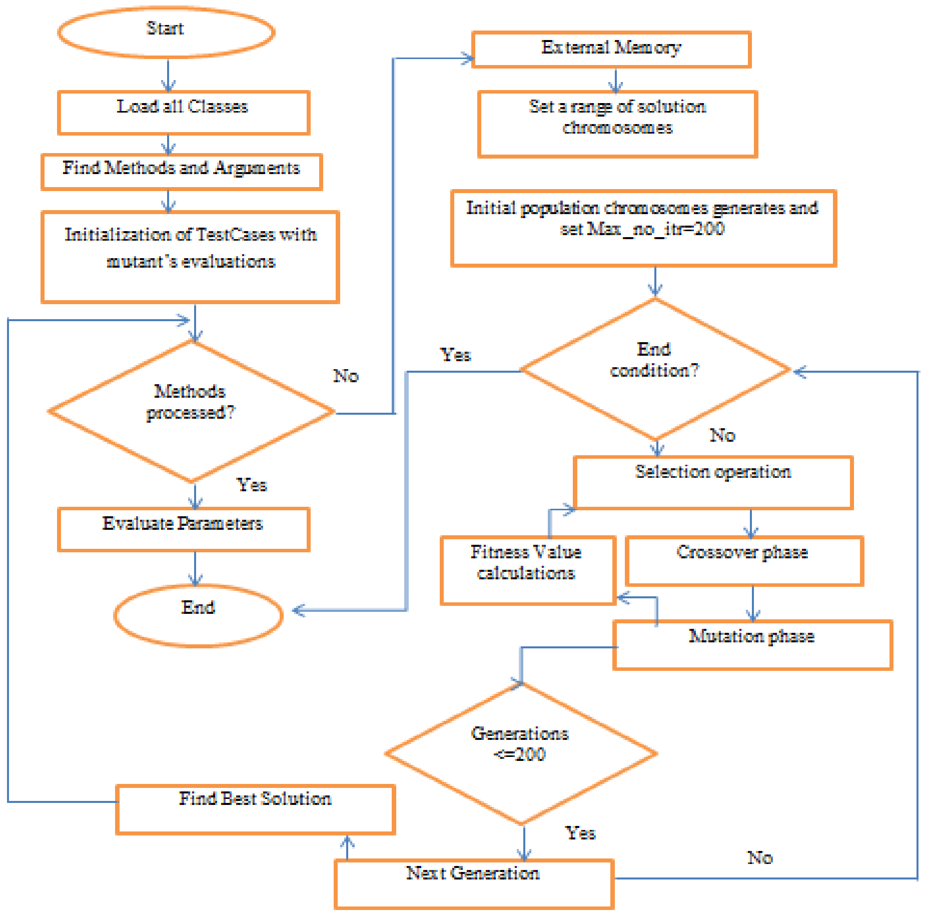
Start
Load all Classes
Find Methods and Arguments
Initialization of TestCases with mutant's evaluations
Methods processed?
No
Yes
Evaluate Parameters
End
External Memory
Set a range of solution chromosomes
Initial population chromosomes generates and set Max_no_itr=200
End condition?
Yes
No
Selection operation
Fitness Value calculations
Crossover phase
Mutation phase
Generations <=200
Yes
No
Find Best Solution
Next Generation

Figure 3. Proposed Flow Diagram

- *Evaluate parameters:* Various parameter mutation score, accuracy, time etc. are used to evaluate perform of the proposed architecture. So at the end of the execution this process is used to find the various parameters to check the performance factor and design a comparison with other existing approaches to performance validations.
- *Stop***:** This process is used to refresh the memory block to create free memory slots and generate the process for further execution.

Various other sub-modules of the optimization process are listed below:-

Step 1: Random Uniformly distribution (Start over the search space). External Memory is an integer valued chromosome (1,0) encoding with direct representation.

Step 2: Assign the fitness function, requiring the number of symmetric coefficients to be optimized for the linear phase, even nth order selected. Manage the upper and lower values of the undefined values as positive 1 and negative 1 respectively.

Step 3: Set the PZ (population size) as 200. Choose a set random solution set of chromosome strings in test cases, with each character consisting of an asset of high selected values.

Step 4: Initialize uniform, select multi-phase operator for random parent selection.

Step 5: Mentioned the best solution at 2, they're being of the next generation and high the error fitness values from fewer values.

Step 6: Crossover and Mutation operators are applied between binary chromosomes to produce off-strings and to prevent redundancy in the resp.

Step 7: generate the test cases by changing the mutation states using different threshold values and perform the change of states in the optimization to evaluate the best appropriate output for the generation to achieve appropriate results.

Step 8: Multistage GA process has been updated. FFn (Fitness Function) is calculated for each data and Less fitval are un-expended at each iteration.

Step 9: Process terminated with the success of fitness or if the high number of generations that is 200 extended earlier.

Exit

Proposed Algorithm in MS-GOA (Multi-Stage Genetic Optimization Algorithm) to solve the travelling salesman problem:-

Initialize the population
Begin with the nearest neighbor heuristics.
For each individual;
Do Johan kabobs (Optional data) individual;
Repeat process
Apply for j=0 to cross over
Then do
Choose two random individuals;
Crossover (Text_cases-chromosomes range divide).
Do Johan kabobs (Optional)-

```
(Child….i.e  sub class, methods and arguments );
    For predefined probability (based on Failure and pass)
    Do mutation (Child); i.e, Mutants (recover the extra spaces and Time).
    Shuffle an individual in the population;
    End;
    Until converged;
    Stop;
```

V. Result and Discussions

In this section is having various results and simulation graphs, processing phases of the test case generation process, and comparisons with various existing algorithms. Here some tabular results of various open source software are used to test the proposed processing architecture and validate their processing and results on all the points. In overall discussion in this section also describe the better performance of test case generation algorithms in terms of accuracy, mutants findings and time consumptions with various test cases.

Table 4. Comparison proposed and existing work in Performance metrics – Accuracy Rate (%)

Project	MGA	GA	BFO	ACO
Banking Software	97.59	90.36	76.15	81.352
Books Management system	98.314	90.138	73.48	86.145
College canteen_System	97.332	90.731	73.135	83.254
Management system_Computer	97.558	92.394	76.95	85.145
Diabetes_checking_system	99.665	92.62	75.158	84.25
Hotel_Management_ System	98.234	92.335	75.158	81.95
LIC_Managements	98.48	92.548	76.552	81.63
Calculation system	98.668	92.145	74.62	84.36
Dictionary_System	99.237	93.885	77.522	85.66
Reserve Tickets_railways	99.35	93.5	70.158	85.24
School_management	98.265	90.54	73.65	87.214
Shuffling_game	97.7	91.88	74.35	81.36
Snake_Game	97.63	93.22	74.214	81.79
Student_Management_System	99.575	93.864	77.95	84.635
Banking_system	99.14	90.913	74.012	87.46
Gaming_system	98.782	90.46	75.28	85.69
Exam_Management	98.99	93.53	75.996	85.21
Job_portal	99.752	93.82	76.42	85.46
Library_system	99.394	98.71	77.25	85.63
Routing_protocol	99.934	98.659	74.25	87.12
Help_ Desk	98.964	92.46	76.235	87.197

Various test cases are generated and recorded in this section for validation of accuracy factor. Here some open source development software codes are collected as dataset and upload in the test case generation system. After processing all this code the system provides accurate factor for the generation one by one. The overall performance of the proposed MGA algorithm as compare to other optimization techniques is better and give better accuracy factor in all the cases.

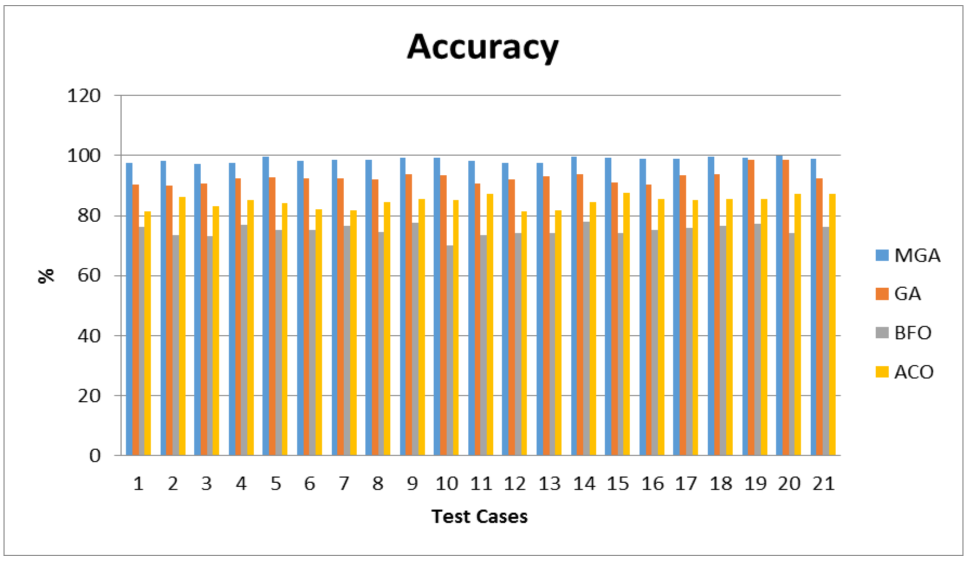

Figure 4. Comparison – Accuracy Rate (%)

As in the table format various test cases are generated on different software systems. All the systems are having their own calculations in the test case generation system with MGA. Here the graphical presentation in a bar chart is shown as a comparison of all the test cases in terms of accuracy factor. The high accuracy shows better results in this system. As in the bar graph, the performance of proposed architecture shows better results than the all other existing approaches. The highest accuracy of the proposed approach crossed 99% for some of the cases.

Table 5: Comparison Between Proposed and existing work in Mutants

Project	**MGA**	**GA**	**BFO**	**ACO**
banking Software	27	17	12	14
Books Management system	31	18	13	17
College canteen_System	35	21	13	16
Management system_Computer	33	19	12	14
Diabetes_checking_system	41	20	14	13
Hotel_Management_ System	45	17	15	17
LIC_Managements	45	16	17	18
Calculation system	42	20	11	15
Dictionary_System	41	19	12	12
Reserve Tickets_railways	41	19	14	15
School_management	27	17	13	14
Shuffling_game	27	19	17	16
Snake_Game	42	18	16	14
Student_Management_System	41	20	14	16
Banking_system	34	20	12	18
Gaming_system	38	20	14	17
Exam_Management	38	20	13	14
Job_portal	34	19	14	15
Library_system	36	19	13	12
Routing_protocol	28	18	14	15
Help_ Desk	41	19	15	16

Mutants are some of the cases which are not passed due to some reasons or development standard. All the mutants degrade the performance of software systems and increase maintenance cost of the developed software system. To reduce the risk of all these things test case generation process takes place. Here in the table performance of MGA is better in all the cases as compare to the other algorithms. The highly accurate system MGA finds mutant accurately more than other existing optimization approaches.

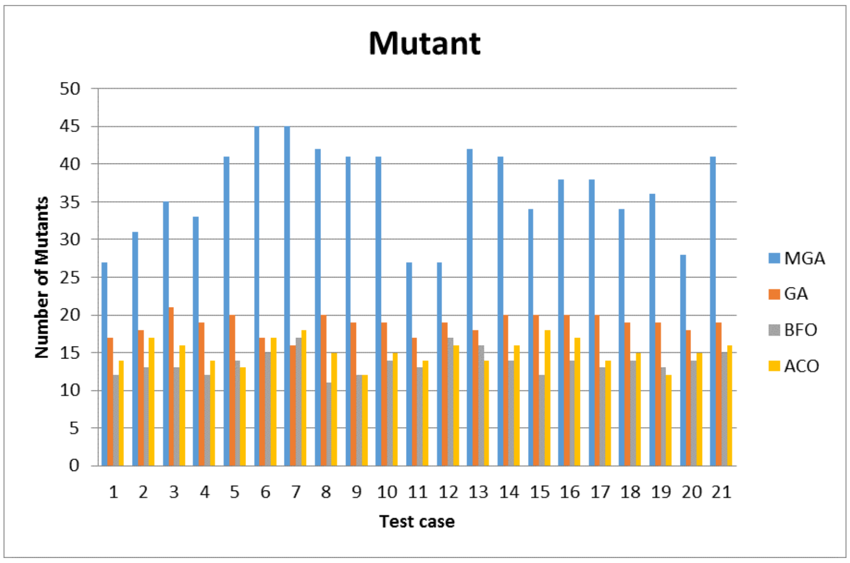

Figure 5. Comparison – Mutant

The figure plots show detected mutants while generating test cases toward various software systems. As the proposed approach MGA perform highly accurate detection for test case generation, so it finds a number of mutants more than other existing approaches. The other existing approach having less accuracy, so that they aren't able to detect the failures occurred in the test case generation process.

Table 6. Comparison between proposed and existing work using - Time Consumption (Ms)

Project	**MGA**	**GA**	**BFO**	**ACO**
banking Software	52	102	85	90
Books Management system	67	94	87	91
College canteen_System	61	97	84	97
Management system_Computer	62	93	83	95
Diabetes_checking_system	61	99	87	96
Hotel_Management_ System	64	102	87	94
LIC_Managements	68	101	89	95
Calculation system	71	94	82	98
Dictionary_System	72	108	89	97
Reserve Tickets_railways	77	107	91	92
School_management	53	107	90	90
Shuffling_game	73	88	84	89
Snake_Game	74	94	86	91
Student_Management_System	55	93	84	94
Banking_system	52	96	87	96
Gaming_system	58	108	89	90
Exam_Management	64	107	94	92
Job_portal	61	107	99	94
Library_system	58	102	98	92
Routing_protocol	52	95	92	98
Help_ Desk	74	97	96	97

The most important part of the software testing is time consuming. The algorithm performs calculated with the help of various parameters like accuracy, error rates, detected mutants, etc. For all the factors time consumption is the most important parameter. Less time taking process is very responsive to the end user to all their queries. Here the test case generation the time consumption of MGA is less as compare to all other existing algorithms. It shows fast processing of uploaded coding modules and provide optimal results in a very short time span.

As in the plotted graph the time consumption of the proposed approach is less as compare to all other existing algorithms. This parameter shows fast processing speed of uploaded code modules and generation of test cases toward them. This parameter helps to process large code modules in less time span and generate highly accurate results corresponding to them.

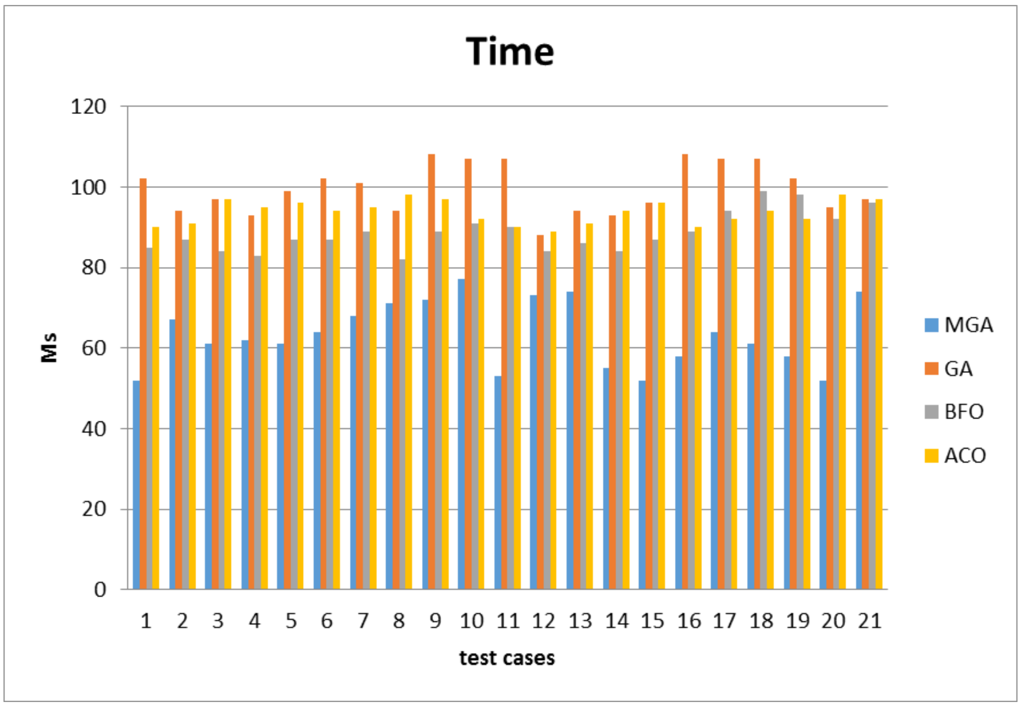

Figure 6. Comparison –Time

VI. Conclusions

The main objective of the research work is depends upon the test cases of object oriented software automatically. There is also an explanation about the applications of evolutionary approaches as genetic algorithm, bacteria foraging optimization algorithm (BFOA) and ant colony optimization (ACO). The research work followed the intelligent approach of multistage Genetic that worked for the huge scale developments of software designs. The entire research work is represented the information in different sections. Software testing is a vital software quality activity to ensure that the benefits of OOPs will be realized. The main objective of software testing is to uncover the as much error as possible with the minimum cost and efforts. A conventional testing process takes place mostly when waterfall life cycle is used for software development at the organization. Nowadays, to overcome the problem which is occurring in conventional testing, object orientation has rapidly become accepted as the preferred paradigm for large scale system design. Object oriented (OO) analysis and design along with agile and other recent software development methodologies leads to object oriented testing. Summarize the software, computing, which includes its classification, the genetic algorithm, BFOA, ACO BCO, PSO, pseudocode, flowcharts and further a comparison analysis of test suite prioritization approaches. In the proposed work, the performance is reliant on the better way. Test case generation is a procedure to generate software corresponding various test case generations and validate various test cases. So that research work identifies the quality of software. This process also declined the maintenance cost (MC) of a software system. In the proposed architecture design, Multi-stage GA (Genetic Algorithm) has various benefits as it is highly effective in higher dimensional spaces, more memory efficient and versatile. Basically, Multi-stage GA is applied in several real-time applications as in the text categorization, classification of test cases and regression related issues. In the proposed approach, mutants compare various existing techniques and performance parameters are like as mutants, accuracy rate, time consumption and number of events. The planned approach is best in terms to enhance the accuracy rate and achieved it in a reduced time period. Several techniques are used to compare the number of events fire. So that, the architecture accuracy rate has achieved this based on the number of events. The Multistage GA test case is an intelligent approach and supportive to various languages like .Net, Java, C++ and Project Management used in an automatic test case. It helps to improve the quality of software and based on the mutants. Basically, mutants are like failure (Some time it is passed or sometimes it fails). The reduced number of mutants increased the software quality.

Future study contains using huge training sets to get better consequences for our proposed approach and also use other benchmark executes for comparing the consequences.

The upcoming work will consider following areas:-

Build a highly effective search path. Because MSGA method is an easy algorithm. It generating effective GA operators find path could significantly enhance test case coverage area and minimize the num_of_iterations.

REFERENCES

1. Binder, R. V. (1994). Testing object-oriented systems: a status report. American Programmer.
2. Kapfhammer, G. M. (2004). Software testing. In In The Computer ScienceHandbook.
3. Ivar, J., Magnus, C., Patrik, J., & Gunnar, O. (1992). Object-oriented software engineering, a use case driven approach. MA.: Addis
4. Dalal, S., & Chhillar, R. S. (2012). Software Testing-Three P'S Paradigm and Limitations. International Journal of Computer Applications, 54(12).

5. Luo, L. (2001). Software testing techniques. Institute for software research international Carnegie mellon university Pittsburgh, PA, 15232(1-19), 19.
6. Blaha, M. (2005). Object-Oriented Modeling and Design with UML: For VTU, 2/e. Pearson Education India.
7. Jalote, P. (2012). An integrated approach to software engineering. Springer Science & Business Media.
8. www.selfgrowth.com/articles/top-10-benefits-of-software-testing)
9. Malik, S., & Nigam, C. (2017). A Comparative study of Different types of Models in Software Development Life Cycle.
10. Pohl, K. (2010). Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated.
11. Davis, A. M., Bersoff, E. H., & Comer, E. R. (1988). A strategy for comparing alternative software development life cycle models. IEEE Transactions on software Engineering, 14(10), 1453-1461.
12. Malik, S., & Nigam, C. (2017). A Comparative study of Different types of Models in Software Development Life Cycle.
13. Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. IJCSI, 5, 95-101.
14. Ruparelia, N. B. (2010). Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, 35(3), 8-13.
15. Bonissone, P. P. (1997). Soft computing: the convergence of emerging reasoning technologies. Soft computing, 1(1), 6-18.
16. Kurhe, A. B., et al., (2011). Soft Computing and its Applications. BIOINFO Soft Computing, Volume 1, Issue 1.
17. Kim, M. Y., & Cheon, Y. (2008, April). A Fitness Function to Find Feasible Sequences of Method Calls for Evolutionary Testing of Object-Oriented Programs. In Software Testing, Verification, and Validation, 2008 1st International Conference on (pp. 537-540). IEEE.
18. Liang, Y. C., & Smith, A. E. (2004). An ant colony optimization algorithm for the redundancy allocation problem (RAP). IEEE Transactions on reliability, 53(3), 417-423.
19. Davidović, T., Teodorović, D., & Šelmić, M. (2015). Bee colony optimization-Part I: The algorithm overview. Yugoslav Journal of Operations Research, 25(1), 33-56.
20. Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on (Vol. 3, pp. 1945-1950). IEEE.
21. Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2016, November). Software Testing Techniques: A Literature Review. In Information and Communication Technology for The Muslim World (ICT4M), 2016 6th International Conference on (pp. 177-182). IEEE.
22. Nouman, M., Pervez, U., Hasan, O., & Saghar, K. (2016, May). Software testing: A survey and tutorial on white and black-box testing of C/C++ programs. In Region 10 Symposium (TENSYMP), 2016 IEEE (pp. 225-230). IEEE.
23. Larrea, M. L. (2017). Black-Box Testing Technique for Information Visualization. Sequencing Constraints with Low-Level Interactions. Journal of Computer Science & Technology, 17.
24. Henard, C., Papadakis, M., Harman, M., Jia, Y., & Le Traon, Y. (2016, May). Comparing white-box and black-box test prioritization. In Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on (pp. 523-534). IEEE.
25. Shin, K. W., & Lim, D. J. (2018). Model-based automatic test case generation for automotive embedded software testing. International Journal of Automotive Technology, 19(1), 107-119.
26. Dalal, S., and Chhillar, R. S. (2012). Case studies of most common and severe types of software system failure. International Journal of Advanced Research in Computer Science and Software Engineering, 2(8).
27. Le Lann, G. (1997, March). An analysis of the Ariane 5 flight 501 failure-a system engineering perspective. In Engineering of Computer-Based Systems, 1997. Proceedings., International Conference and Workshop on (pp. 339-346). IEEE.
28. Gray, J., and Siewiorek, D. P. (1991). High-availability computer systems. Computer, 24(9), 39-48.
29. Rahideh M, Mazloum SZ. Combination System Optimization of Solar Collector/ Photovoltaic with Genetic Algorithms. Medbiotech Journal. 2019;03(02):58-64.
30. Hosseini Naghavi AB, Alishah O, Gorji AM. Investigation of the Genetic Diversity of Cultivars and Lines of Tetraploid Cottons by the use of Quantitative Morphologic Properties of Fibers. Medbiotech Journal. 2019;03(02):65-9.
31. Amanlou M, Mostafavi SM. In sillico screening to aim computational efficient inhibitors of caspase-9 by ligand-based pharmacophore modeling. Medbiotech Journal. 2017;01(01):34-41.
32. Mostafavi SM, Bagherzadeh K, Amanlou M. A new attempt to introduce efficient inhibitors for Caspas-9 according to structure-based Pharmacophore Screening strategy and Molecular Dynamics Simulations. Medbiotech Journal. 2017;01(01):1-8.
33. Salehi, S., & Mo'tadel, M. (2015). Model presentation to feasibility measurement of knowledge management implementation with ANP approach (case study of Post bank) . *UCT Journal of Research in Science, Engineering and Technology*, *3*(4), 17-23.
34. Freitas, M. D. C., & Mira da Silva, M. (2018). GDPR Compliance in SMEs: There is much to be done. *Journal of Information Systems Engineering & Management*, *3*(4), 30.
35. Mendoza, D. J., & Mendoza, D. I. (2018). Information and Communication Technologies as a Didactic Tool for the Construction of Meaningful Learning in the Area of Mathematics. *International Electronic Journal of Mathematics Education*, *13*(3), 261-271. https://doi.org/10.12973/iejme/3907

AUTHORS PROFILE

Anju Bala, Department of Computer Science and Applications Maharishi Dayanand University, Rohtak; Email: anjunarwal024@gmail.com

ssssssssssRajender Singh Chhillar, Department of Computer Science and Applications Maharishi Dayanand University, Rohtak; Email: Chillar02@gmail.com