FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X

Optimization of Software Quality Attributes using Evolutionary Algorithm

Priyanka Makkar[1], Sunil Sikka[2] and Anshu Malhotra[3]

[1]Research Scholar, Department of Computer Science, Amity University Haryana, India, priyanka.makkar@rediffmail.com
[2]Associate Professor, Department of Computer Science, Amity University Haryana, India, ssikka@ggn.amity.edu
[3]Associate Professor, Department of Computer Science, The NorthCap University, Gurugram, India, anshumalhotra@ncuindia.edu

***Corresponding Author:** Priyanka Makkar; E-mail: priyanka.makkar@rediffmail.com

ABSTRACT- Software quality is a multidimensional concept. Single attribute can't define the overall quality of the software. Software developer aims to develop software that possesses maximum software quality which depends upon various software quality attributes such as understand ability, flexibility, reusability, effectiveness, extendibility, functionality, and many more. All these software quality attributes are linked with each other and conflicting in nature. Further, these quality attributes depend upon the design properties of the software. During the designing phase of software, developers must optimize the design properties to develop good software quality. To obtain the appropriate value optimization is done. This paper implemented two multi-objective evolutionary algorithms (NSGA-2 and MOEA/D) to optimize software design properties to enhance software quality. While comparing NSGA-2 algorithm with original values it is found that there is a 1.73% improvement in the software quality on the other hand MOEA/D shows a 3.58% improvement in the software quality.

Keywords: Software Metrics; Software Quality; Software Quality Attributes; QMOOD.

ARTICLE INFORMATION

Author(s): Priyanka Makkar, Sunil Sikka and Anshu Malhotra ;

Special Issue Editor: Dr. Vikash Yadav;
Received: 30/03/2022; **Accepted**: 11/05/2022; **Published**:22/05/2022;
E- ISSN: 2347-470X;
Paper Id: 0422SI-IJEER-2022-06;
Citation: 10.37391/IJEER.100214
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-10/ijeer-100214

This article belongs to the Special Issue on **Recent Developments in Communication Technology using Machine Learning Techniques**.

Publisher's Note: FOREX Publication stays neutral with regard to jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

Dependencies on the software are increasing day by day. From time-to-time software requires regular updates and modifications to satisfy the customer needs. While updating the software it is essential to focus on the quality. Poor quality of the software has serious consequences such as it may cause loss of life, mission failure, financial loss, and permanent injury. To ensure the quality, measuring software quality is essential. In literature, several quality models are available to measure the software quality, and a few such models are, Boehm's Quality Model [12], FURPS Quality Model [19], Dromey's Quality Model [16], ISO 9126 Quality Model [10], and Quality Model for Object-Oriented (OO) Design (QMOOD) [15]. Out of all these models, this research work focuses on QMOOD hierarchical model to analyze the quality of software. QMOOD considers both internal as well as external quality attributes of the software. As well as it also maps source code metrics to a higher abstraction level. Researchers have used various metrics to measure these attributes. All these quality attributes further depend upon ten design properties. These design properties can be measured with the help of software metrics. These metrics are Depth of Inheritance (DIT), Response for class (RFC), Number of Children (NOC), Coupling between Objects (CBO), Weighted Method per Class (WMC), Lack of Cohesion of Methods (LCOM). These six metrics suites are known as Chidamber & Kemerer (CK) metrics [17] and four other OO metrics are Number of Attributes, Number of Method, Number of Private Attributes, and Number of public methods. These metrics are a decent indicator to describe the software quality during the design phase. The objective of this research is to optimize the design properties of the software by using the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) and Non-Dominated genetic algorithm (NSGA-2) algorithms. Software developers can use these optimized values of design properties while updating the software it will enhance the software quality.

This paper covers background study in section II; explanation of QMOOD in Section III, research methodology is described in Section IV. The proposed model is applied to QMOOD to maximize software quality, and this is explained in Section V. The final section concludes the paper and highlights the future scope.

2. BACKGROUND STUDY

In literature various techniques has been used by many researchers to enhance the software quality such as fault prediction, refactoring, class prioritization, optimization of software quality attributes. Author R. Malhotra et al. [9] emphasis the importance of software quality attributes and proposed a model used QMOOD for fault prediction at early stage of software life cycle by using statistical method and

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X

machine learning for improving software quality attributes. And it is found that metrics WMC and LOC are help in predicting the software quality. Author Puneet at al. [8] develop a model use to evaluate the quality of java program based on QMOOD. Author gave input as java program which vary in functionality and complexity, and it was concluded that Total Quality Index (TQI) of software having single class is very less whereas program which are having low complexity and high design metrics have high TQI. Usman Mansoor et al. [5] proposed the novel approach that enables software developers to apply refactoring technique on model level. This technique improves the inter diagram consistency, as well as overall software quality. For model-based technique UML based class diagram and activity diagram are used to improve software design quality by applying NSGA-II and MOEA/D multi-objective evolutionary algorithms. Authors also found the best tradeoff between multiple criteria and suggested feasible and good design refactoring solution. The results of the proposed approach were statistically examined on four open-source software. Salazar et al. [13] present 3 multi-Objective reliability optimization problems first to find the optimal number of redundant components second is to find the reliability of the components and third is to identify both reliability and redundancy. These issues were stated as single objective mixed-integer non-linear programming problems with one or more constraints. These challenges are framed as multiple-objective problems (MOPs) and were solved with a second-generation Multiple-Objective Evolutionary Algorithm (MOEA) that handles constraints. The Pareto front, or set of optimal solutions, as well as optimal solution space, are found using NSGA-II. Ouni et al. [7] proposed a technique based on multi objective optimization approach to find the optimum trade-off between semantic coherence and software quality using NSGA-II. Two criteria, vocabulary similarity and structural coupling, are used to achieve semantic coherence. The proposed approach's helps in the development of more meaningful refactoring solutions that addressed design flaws while maintaining semantic coherence. From the survey it is observed most of the work has been done on statistical method, machine learning. Evolutionary Algorithms (EA) are another approach of optimization which has gained lot of popularity among researcher and software developers. In this research EA are used to optimize the software quality.

3. Quality Model for Object-Orieted Design (QMOOD)

In 2002 Bansiya and Davis proposed the model QMOOD [15], which measures the quality of software in terms of attributes like reusability, functionality, flexibility, extendibility, understand ability, and effectiveness. QMOOD explains the relationship between Qualities Attributes (QA) with design properties like abstraction, encapsulation, and Coupling. *Figure 1* represents the QMOOD. It implies four levels L_1: design quality attributes, L_2: OO design properties, L_3: OO design metrics, L_4: OO design components, and three mappings' links L_{12} linking design property to quality attributes, L_{23} linking design metrics to design properties, and L_{34} linking OO design component to design properties.

Table 1 represents the relationship between design properties and QA. External quality attributes are computed with these equations designed by Bansiya et al. [15] according to the author metrics that are used to access design properties can be changed or a distinct set of design properties can be used. Therefore, in *Table 2* mapping of design properties with software design metrics according to the dataset is justified.

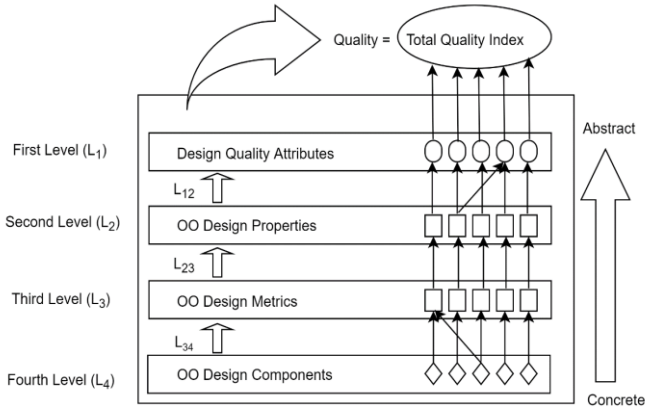


Figure 1: Quality Model for OO Design [15]

Table 1: Computation Equation for QA [15]

QA	Computation Equation
Understandability	-1/3 *Abstraction + 1/3*Encapsulation -1/3*Coupling+1/3*Cohesion-1/3* Polymorphism -1/3* Complexity -1/3*Size
Functionality	+3/25*Cohesion + 11/50 * Polymorphism + 11/50 * Messaging +11/50*Size + 11/50 * Hierarchies
Extendibility	+1/2*Abstraction-1/2* Coupling +1/2*Inheritance +1/2*Polymorphism
Effectiveness	+1/5*Abstraction + 1/5* Encapsulation+1/5*Composition +1/5*Inheritance +1/5*Polymorphism
Reusability	-1/4*Coupling + 1/4* Cohesion + 1/2* Messaging +1/2*Size
Flexibility	+1/4*Encapsulation -1/4* Coupling +1/2*Composition +1/2*Polymorphism

Table 2: Mapping of Design Properties to Software Design Metrics

Design Property	QMOOD [Design Metric]	Equivalent Metrics Computed	Description
Design size	Number of Methods	Class	Total number of Classes
Complexity	Direct Class Coupling	WMC	Weighted Method per Class
Coupling	Design Size in Classes	CBO	Coupling Between Objects
Polymorphism	Average Number of Ancestors	NOM	Number Of Method Range (0 to 1) Max=0 Min=NOM
Hierarchies	Cohesion among Methods of Classes	NOC	Number Of Children
Cohesion	Measure of Aggregation	LCOM	Lack of Cohesion of Methods
Abstraction	Data Access Metrics	1/DIT	1/ Depth of Inheritance Tree

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X

Encapsulation	Number of Hierarchies	NOPA	(No of Private attribute/No of Attribute)
Composition	The measure of Functional Abstraction	NOA	NOA Range (0 to 1) Max=0 Min=NOA
Inheritance	Class Inheritance Size	DIT	Depth of Inheritance Tree
Messaging	Number of Polymorphic Methods	RFC	response for a Class /Number of public method

4. RESEARCH METHODOLOGY

In the late 1960s, Genetic Algorithms (GA) was introduced by John Holland [18]. GA help in evaluating the complicated optimization problem, which cannot be solved using mathematical techniques. GA evaluates on randomly generated initial population which contains solutions also called individuals. These individuals or chromosomes are used to calculate their fitness value or objective function of the problem. It represents the quality of the solution.

In this research paper, we are optimizing the quality of software having six objective function reusability, functionality, flexibility, extendibility, understandability, and effectiveness called fitness function. The fitness value depends upon the quality of the solution.

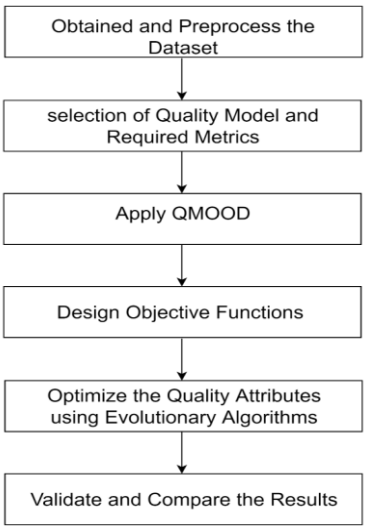


Figure 2: Research Methodology of Proposed Work

Figure 1 represents the complete workflow of proposed technique. First, obtained and preprocess the dataset. Select the quality model as well as metrics. Apply QMOOD to the selected dataset and design the objective functions. Optimize the quality attributes and calculate the six objective functions using two evolutionary algorithms. Finally, validate and compare the result.

4.1 Dataset and Parameter Setting

Open-source dataset Eclipse Java Development Tool (JDT) Core is chosen for this research. This dataset has 1041 classes, 91 software versions, 23 metrics are available out of these 10 metrics are used in this research work. Table 3 represents the detail of parameter settings that have been taken to perform these experiments. The performance of GA depends upon its control parameters [3-4].

Table 3: Experimental Parameter Settings for modify MOEA/D and modify NSGA-2

Type	Description
Type of genetic algorithm	Multi-Objective Evolutionary Algorithm
Multi-objective approach	Ideal multi-objective
Type of Decision Variables	Continuous
Type of Optimization	Linear
Population size (Total no. of classes)	1041
Representation of chromosome	Real coding
Selection operator	Tournament selection method
Tournament size	2
Termination criteria	Max iteration (100)
Size of the initial population (Parent Population)	50 chromosomes (~5% of the total population)
Q in MOEA/D (Size of Neighbors)	8(0.15% of the initial population)
Crossover operator	Simulated Binary crossover
Crossover Rate	0.2
Mutation Rate	0.02

4.2 Optimization

An optimization algorithm is a process that compares numerous solutions iteratively until an optimum or satisfying result is obtained. To obtain optimized results various optimization algorithms are available. In this research work two evolutionary algorithms MOEA/D and NSGA-2 are used to optimize the fitness function.

4.2.1 Steps of MOEA/D

Algorithm-1 Modify MOEA/D for an optimized class level quality parameter of the software

Input: *Data_set*: Class name and class quality attribute values of multiple versions.
Stopping Condition: Number_of_Iterations
P: Sub-problems to consider.
Q: No of the neighbours for each *P* sub-problems.
$\mu^1, \mu^1, \ldots \mu^P$: Uniformly distributed weight vectors
EP: External Population z_i: Initial Population
Output: Classes having optimized quality attribute
Initialization

Step 1- Set EP = {Ø}, (Store all Non-dominated solutions in EP.)

Step 2- μ^i, for each uniformly distributed weight element, calculate the Q nearest weights, $\mu^{i(1)}, \mu^{i(2)} \ldots \mu^{i(Q)}$ using the Euclidean distance and set neighborhood
$\sigma(i) = \{i(1) \ldots . i(Q)\}$.

Step 3- Produce the initial population $z_1 \ldots z_p$ and evaluate the fitness of each population using the objective function $f_u(z_j)$. (Here, the fitness functions are six quality attributes of QMOOD.)

Step 4- Set $\alpha = (\alpha_1 \ldots . \alpha_m)$ where α is the reference point.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X

Step 5- Repeat for Number_of_iterations

5.1 For each Sub problem under MOP l =1 to P, do

5.2 Reproduction: Produce newer solutions $\boldsymbol{\pi}$ using individuals z_u and z_v with crossover operator, here one point cross**over operation is applied, for** $\boldsymbol{u, v} \in \boldsymbol{\sigma(l)}$.

5.3 **Update of** $\boldsymbol{\alpha}$: for $r = 1, \ldots, m$, if $f_j(\boldsymbol{\pi}) < \alpha_r$ set $\alpha_r = f_r(\boldsymbol{\pi})$

5.3.1 **Update neighboring solutions**: For each $k \in \boldsymbol{\sigma}(i)$, if $g(\boldsymbol{\pi} \mid \mu^k)$, then set $z_k = \boldsymbol{\pi}$ and $f_r(z) = f_r(\boldsymbol{\pi})$, $r = 1 \ldots m$.

5.3.2 **Update of External Population**: Eliminate dominated solutions $\boldsymbol{\pi}$ from ***EP*** if $\boldsymbol{\pi}$ not dominated by any member in EP, add $\boldsymbol{\pi}$ to EP.

Step 6- Output ***EP***

MOEA/D was proposed in 2007 by Li and Zhang [11]. MOEA/D decomposes the multi-objective problem (MOP) into several sub-problems rather than solving a problem as a whole and solves all these N sub-problems concurrently by providing solutions in terms of the population. It has less computational complexity at each generation and improved decomposition approaches than NSGA-2, NSGA-3, and multi-objective local search (MOLS). MOEA/D uses a small population size and can produce very evenly distributed solutions of a small number. MOEA/D is applied to optimized class quality parameters. First, the data set is imported having the class name and class quality attribute values for multiple versions then creates the sub-problem. MOEA/D randomly selects 50 classes that build the initial population of classes from the imported data which is sufficient to calculate the quality of complete software. There is no rigid rule for selecting the initial population size. The size of the random population determines by the programmer. If the size is large, it comes with a quick or better solution but will cost more time. On the other hand, if a population's size is less, it quickly terminates the solution but may not give the correct result always and affect the overall working of GA. After initializing the population, the fitness function with the help of QMOOD is calculated. Determine the decomposed cost of selected classes in the population takes the maximum cost value from the population. Calculate the domination of the population. This loop will run until the maximum iteration, and this is the exit criteria. Run the loop for i= 1 to nPop (50). Select two random class versions from sub-problems, called positions as per QMOOD [1-2]. Apply single-point crossover between selected positions and generate a new chromosome.

Figure 3 represents the flow of MOEA/D. Calculate the cost (QMOOD) for this new chromosome. Update the variable having maximum cost, then calculate the decomposed cost for this new chromosome.

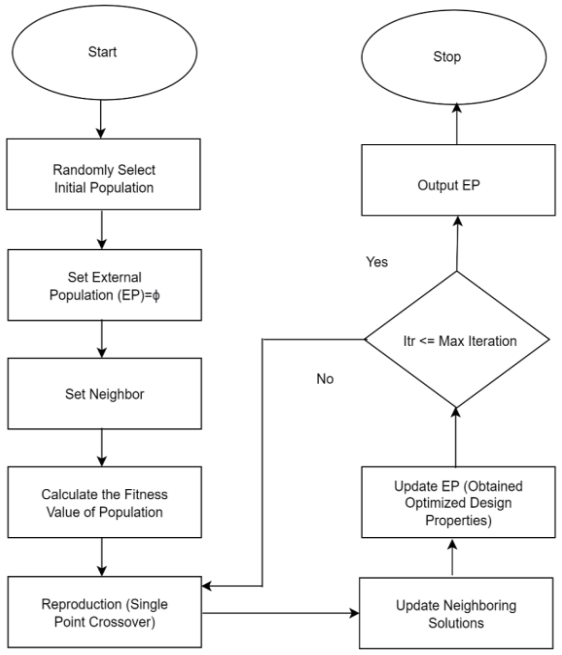


Figure 3: Framework of MOEA/D

Run the loop for j=1 to max sp(i) neighbors. Replace the current class version in the population with a newly created chromosome if the decomposed cost is better than the current determining domination status of the population. Finally, in the end, a pseudo version of classes having optimized quality attributes and parameters is formed.

4.2.2 Steps of NSGA-2

NSGA-2 was proposed by Deb et al. [14] in the year 2002. *Figure 2* represents the working of NSGA-2 algorithm. This algorithm is applied to the same randomly selected classes. In NSGA-2 first initialize the population then sorting takes place based on the non-domination criteria of the population. Once the sorting is complete front number is assigned to the solution. The individuals in the population are selected based on front rank and those solution having same front are selected on the basic of crowding distance. The selection of individuals is carried out using a binary tournament selection with the crowded-comparison operator. Real coded GA using simulated binary crossover and polynomial mutation. The offspring population and current generation population are combined, and the individuals of the next generation are set by selection. The new generation is filled by each front subsequently until the population size exceeds the current population size.

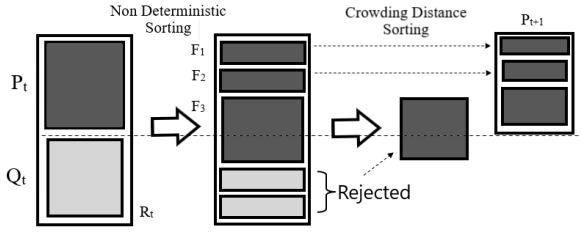
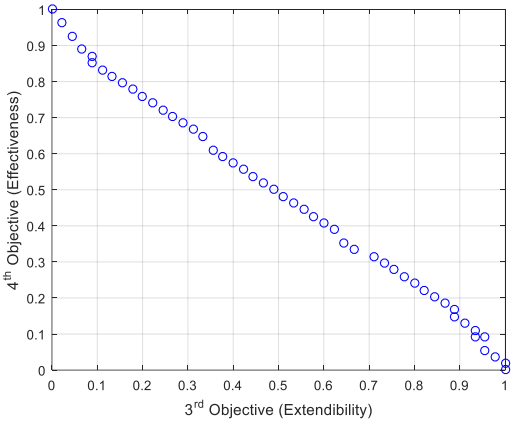


Figure 4: Non-Dominated Sorted Genetic Algorithm-II [9]

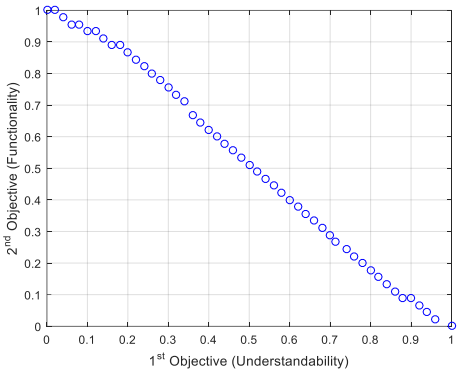
FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X

5. SIMULATION RESULTS

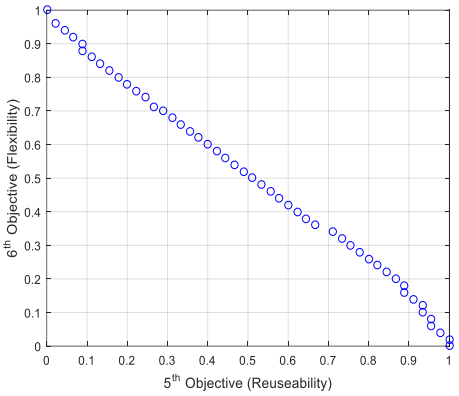
The proposed technique has been carried out on an Intel(R) Core (TM) i5-2430M Processor running at 2.40 GHz with RAM of 4 GB. MATLAB R2016a is used to implement algorithms. The solutions obtained over100 runs are optimal and non-dominated after that these optimized values are normalized between zeros to one. The modified MOEA/D and modified NSGA-2 algorithm return multiple optimal solutions because of the random selection procedure of this algorithm some time multiple solutions have similar objective values but for brevity, all the similar solutions except one have been removed and graphs are generated between them. The dots or small circles represent the Pareto optimal solutions obtained from modify MOEA/D modified NSGA-2. As the problem is maximization, the points which are farthest from the axis are to be considered. *Figures 5 and 6* represents the graphs obtained from modify MOEA/D and modify NSGA-2. *Figures 5 and 6 (A)* represents the graph between objective function 1 (Understandability), and objective function 2 (Functionality). *Figures 5* and 6 (B) represents a graph between objective function 2 (Extendibility), and objective function 4 (Effectiveness). *Figures 5 and 6 (C)* represents a graph between objective function 5 (Reusability), and objective function 4 (Flexibility) the graphs are called Pareto front. The implementation result is summarized in *Table 4*. It represents the total optimized values of all six objective functions obtained after applying the modified MOEA/D and modified NSGA-2 algorithms on fifty classes. It eliminates dominated solutions and produces newer solutions from the external population if newer solutions are not dominated by any member in EP, and adds newer solutions to EP. After adding all the quality attributes of six objective functions of a class the Total Quality Index (TQI) of the class is obtained. It is found that the TQI of the software after applying to modify MOEA/D algorithm gets increased. *Table 5* represents the quality improvement in percentage. While comparing modify NSGA-2 algorithm with original values it is found that there is a 1.73% improvement in the software quality on the other hand modify MOEA/D shows a 3.58% improvement in the software quality.



(A)

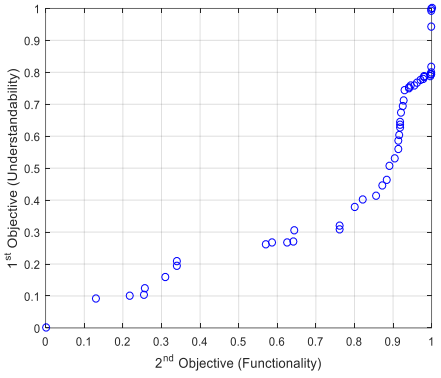


(B)

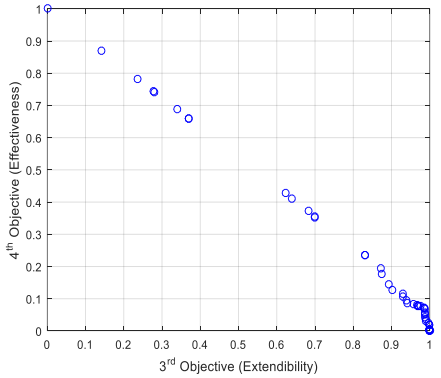


(C)

Figure 5: Pareto Optimal solutions obtained from modify MOEA/D on Eclipse JDT Core dataset between (A) Understand ability and Functionality (B) Extendibility and Effectiveness (C) Reusability and Flexibility with their Optimal Normalized value

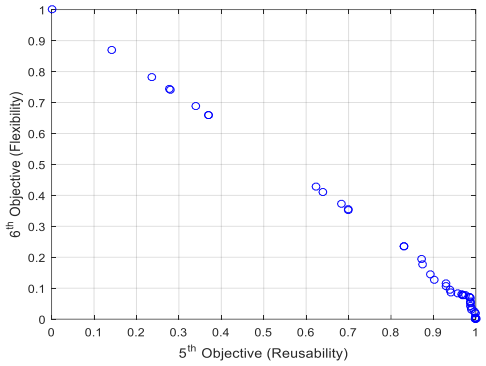


(A)



(B)

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X



(C)

Figure 6: Pareto Optimal solutions obtained from modify NSGA-2 on Eclipse JDT core dataset between (A) Understand ability and Functionality (B) Extendibility, and Effectiveness (C) Reusability and Flexibility with their Optimal Normalized value

In this research work, modify MOEA/D and modify NSGA-2 algorithms are used to optimize the quality parameter and obtain improved quality parameters. The work presented as well as the dataset has not been explored by any of the researchers. Both the algorithms have been modified to work with the dataset used in this research. It has been found that modified MOEA/D gave a better TQI value of a class as compared to NSGA-2 and original data. After having the optimized values of class parameters, it is easy for software engineers to work on those parameters, which overall degrade the software quality. It can easily identify which class has low-quality attributes in terms of functionality, reusability, effectiveness, understandability, extendibility, and flexibility. So that, while updating the next version of the software, the developer will keep in mind which class has low-quality value.

Table 4: Quality improvement in modify MOEA/D and modify NSGA-II w.r.t original values

QA	Modify MOEA/D	Modify NSGA-2	Original values
Understand ability	27826	27802	27678.71
Functionality	257.83	210.33	160.5488
Extendibility	3451.38	3360.13	3272.459
Effectiveness	12780.5	12279.51	11889.93
Reusability	272.78	185.59	140.06
Flexibility	273.79	224.67	168.4355
TQI	44862.24	44061.52	43310.15

Table 5: Quality Improvement of Software Attributes in percentage

QA	Modify MOEA/D	Modify NSGA-2
Understand ability	0.532	0.44
Functionality	60.59	31.00
Extendibility	5.46	2.67
Effectiveness	7.49	3.27
Reusability	94.75	32.50
Flexibility	62.54	33.38
TQI	3.58	1.73

5.1 Comparative Analysis

Author Mandeep et al. [8] also calculated the quality attributes using the QMOOD quality model. By capturing the object-oriented software metrics of eclipse plugin tool Metrics 1.3.6 metrics scores are normalized but no genetic algorithm is used. Mansoor et al. [5] apply refactoring technique to improve software quality for this both MOEA/D and NSGA-2 algorithm were used on QMOOD, and it was found that functionality of the software get decreases.

;

6. CONCLUSION & FUTURE SCOPE

In this paper, modify MOEA/D and modify NSGA-2 algorithms provide the optimized values of class design properties through which external quality, as well as TQI of the software, get improved. And it is found modify MOEA/D shows better results. The range of improvement of each class lie between 1to 5% and overall improvement in the software is found 3.58%. It also provides a population of the class having an optimized value of class quality attributes. After having the optimized values of class parameters, it is easy for software engineers to work on those parameters, which overall degrade the software quality. It can easily identify which class has low-quality attributes in terms of functionality, reusability, effectiveness; understand ability, extendibility, and flexibility. So that while updating the next version of the software developer will keep in mind which classes have low-quality parameters.

In the future, more work can be done by adding more quality attributes. The proposed technique is computed using static features of the software however the same work can be further extended for dynamic features and for software engineers to work on those parameters, which overall degrade the software quality. It can easily identify which class has low-quality attributes in terms of functionality, reusability, effectiveness, understand ability, extendibility, and flexibility. So that, while updating the next version of the software, the developer will keep in mind which class has low-quality value can be further considered for complex and large real-time software.

REFERENCES

[1] Karakonstantis, Ioannis, and Aristidis Vlachos. "Bat algorithm applied to continuous constrained optimization problems." Journal of Information and Optimization Sciences 42, no. 1, pp-57-75, (2021).
[2] Makkar, Priyanka, Sunil Sikka, and Anshu Malhotra. "A Multi-Objective Approach for Software Quality Improvement." Journal of Physics: Conference Series. Vol. 1950. No. 1. IOP Publishing, 2021.
[3] Indu, and Rishipal Singh. "Trajectory planning and optimization for UAV communication: a review." Journal of Discrete Mathematical Sciences and Cryptography 23.2 (2020): 475-483.
[4] Torre, Ennio, et al. "A dynamic evolutionary multi-objective virtual machine placement heuristic for cloud data centers." Information and Software Technology 128 (2020): 106390.
[5] Mansoor, U., Kessentini, M., Wimmer, M., & Deb, K. (2015). Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm. *Software Quality Journal*, *25*, 473-501.
[6] Goyal, Puneet Kumar, and Gamini Joshi. "QMOOD metric sets to assess quality of Java program." 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). IEEE, 2014.
[7] Ouni, A., Kessentini, M., Sahraoui, H., & Boukadoum, M. (2013). Maintainability defects detection and correction: a multi-objective approach. *Automated Software Engineering*, *20*(1), 47-79.

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 10, Issue 2 | Pages 131-137| e-ISSN: 2347-470X

[8] Chawla, Mandeep K., and Indu Chhabra. "Capturing OO Software Metrics to attain Quality Attributes-A case study." International Journal of Scientific & Engineering Research 4.6 (2013): 359-363.
[9] R. Malhotra and A. Jain, "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality," Journal of Information Processing Systems, vol. 8, no. 2, pp. 241–262, Jun. 2012.
[10] Al-Qutaish, Rafa E. "Quality models in software engineering literature: an analytical and comparative study." Journal of American Science 6.3 (2010): 166-175.
[11] Zhang, Qingfu, and Hui Li. "MOEA/D: A multiobjective evolutionary algorithm based on decomposition." IEEE Transactions on evolutionary computation 11.6 (2007): 712-731.
[12] Boehm, Barry W., J. R. Brown, and M. Lipow. "Quantitative evaluation of software quality." Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research 69 (2007): 21.
[13] Salazar, D., Rocco, C. M., & Galván, B. J. (2006). Optimization of constrained multiple-objective reliability problems using evolutionary algorithms. *Reliability Engineering & System Safety*, *91*(9), 1057-1070.
[14] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197.
[15] Bansiya, Jagdish, and Carl G. Davis. "A hierarchical model for object-oriented design quality assessment." IEEE Transactions on software engineering 28.1 (2002): 4-17.
[16] Dromey, R. Geoff. "A model for software product quality." IEEE Transactions on software engineering 21.2 (1995): 146-162.
[17] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design." IEEE Transactions on software engineering 20.6 (1994): 476-493.
[18] Holland, John H. "Genetic algorithms." Scientific american 267.1 (1992): 66-73.
[19] Grady, Robert B. Practical software metrics for project management and process improvement. Prentice-Hall, Inc., 1992.