# Enhancing Software Requirements Selection Process Using Genetic Algorithm

Mohd. Nazim[1], Javed Ahmad[2], Warda Farhan[3], Md Waris Ansari[4], Saba Hoda[5], Varishth Bhaskar[6]
{mohdnazim@jamiahamdard.ac.in[1], javed@jamiahamdard.ac.in[2], farhan.warda337@gmail.com[3]}

Department of Computer Science and Engineering, SEST, Jamia Hamdard, New Delhi[1,2,3]

**Abstract.** Software requirements selection is an important phase of the software development process. It is very difficult to decide the most important software requirements from a large set. Therefore, the capability of the genetic algorithm to assist in making better decisions on the software requirements selection is demonstrated in this paper using a practical case study. The objective of the genetic algorithm implemented in the software requirements selection function is to get a better decision in selecting the appropriate software requirements based on a range of criteria in various circumstances. To acquire information on assessment criteria and requirements, we used an Institute Examination System. The results obtained from experiments based on different criteria by prioritizing different software requirements have proved the capability of genetic algorithm to select the best solution. Our proposed genetic algorithm based software requirements selection approach is compatible with existing genetic algorithm based techniques and as well as the some other fuzzy based techniques also.

**Keywords:**Crossover, Fitness Value, Genetic Algorithm, Institute Examination System, Mutation, Software Requirements Selection.

## 1 Introduction

The Software Requirements (SRs) Selection is the vital phase of a software development process [1]. SRs selection typically involves assessing the performance and quality of SRs based on a list of criteria. When there are a large number of SRs to consider, making the decision on which ones to include in the first version of the product becomes much more difficult. We proposed developing a smart function based on a Genetic Algorithm (GA) to facilitate the decision-making process for the SR selection. The GA is an optimization method that is based on Darwin's theory, *"only the fittest survive"* [2]. GA can identify solutions to problems without any biases because it just requires the measure of fitness for a setup in the space of solutions [3].

Practically it is not possible to consider all the SRs in a single release of software because of many constraints of a company like cost, manpower, time, etc. [4,19]. So, this work is presented through a practical approach in which GA is applied in a function to make a better decision on the selection of SRs. For evaluation, we used a dataset of functional requirements (FRs) of an Institute Examination System (IES) proposed by Nazim *et al*. [5]. We used three non-functional requirements (NFRs) as the evaluation criteria to evaluate the FRs.

ICIDSSD 2022, May 24-25, New Delhi, India

DOI 10.4108/eai.24-5-2022.2319030

Results collected from studies have proved the GA's potential to choose the "fittest" solution depending on different situations and by prioritizing various evaluation parameters. The following is how the work is structured: Section 2 gives background information and a literature review, Section 3 explains the methodology, Section 4 presents a case study with experimental data analysis, and Section 5 presents the conclusion and the suggestion for future research work.

## 2 Literature review

A large number of studies and researches that used GA are available in the literature. For example, a GA-based feature subset selection framework is proposed by [6] that can accept multiple feature selection criteria and identify the small clusters of features. In a study in [7], a tool named SCOUT is proposed for the conceptual spacecraft design by using the concept of GA. [8] proposed a *"genetic algorithm feature selection (GAFS) for the image retrieval systems and image classification"*. A method for the selection of seismic attributes by using the GA approach is proposed by [9] that uses the neural network training outcomes for the selection of the best type and number of seismic attributes. [10] developed a decision-making model for selection of supplier by using neural networks GA, in which GA is used initialize the weights for the network. A hybrid algorithm based on GA and ant colony optimization algorithms is developed by [11] for the solution of a multi-criteria supplier selection problem. In a study [12], the GA is used with the Bayesian approach for the green supplier selection. A GA-based method to optimize the efficiency of the software testing process was proposed by [13], and this method focuses on the most critical paths in a program. In our literature review, we have not found any study that used a genetic algorithm for the SRs selection; therefore it motivated us to develop a function by using GA to select the high ranked SRs from a large dataset based on various criteria. The high ranked SRs selection helps in decreasing the production time, manpower, overall cost, i.e., for the development of any software product, while it increases the efficiency of a software product.

## 3 Methodology

Various activities are involved in our approach like determining the selection criteria set, building a selection function utilizing GA for an ideal solution, computing fitness values, generating an initial population, selecting individuals, performing crossover operation over the parents, and mutation as well. The next sections go over these activities in detail.

### 3.1 Determining a Set of Selection Criteria

The establishment of selection criteria set is the initial stage in any SRs selection process. There is a wide range of such criteria as security, reliability, usability, etc. Security is the most important aspect of software functionality because there is a plethora of evidence of financial loss due to the use of insecure software systems [14]. *"It is a non-functional requirement and it has typical security properties like confidentiality, integrity, availability, accountability and access control"* [15]. Reliability can be defined as the possibility of the function/requirement sustaining for a particular period of time in a particular environment [16]. Reliability has been

identified as the most important quality measure to determine the success of a software project. Usability is concerned with how to make a requirement or a function system work properly. Considering usability throughout the requirements gathering phases means the same thing as defining a software quality attribute early in the development process [17].

### 3.2 Using a GA

The GA is a computational searching method for finding exact or approximate solutions to the search issues.GA is a type of evolutionary algorithm that employs concepts including inheritance, mutation, selection, and crossover that are based on evolutionary biology.

The process flow steps of a GA are as given below:

Step 1: Create an initial population at random.
Step 2: Evaluate all individuals of the population using the fitness function.
Step 3: By using the roulette selection, select the fittest individuals as parents (i.e., parent-1 and parent-2) from the population.
Step 4: perform the crossover operation over parent-1 and parent-2 to form a new child (or chromosome). Check the fitness value of a new child to see whether the mutation is needed or not?
Step 5: Steps 3–4 are continued until a new population is formed.
Step 6: Steps 3-5 are repeated over 5 generations.
The diagrammatical representation of the steps of a GA method is shown in Fig. 1.

Typically, GA evolution begins with a population of randomly produced individuals (chromosomes). The fitness value of all individuals of a population is computed in each generation. On the basis of the fitness values, some individuals are selected from the current population and then modified, recombined, and probably mutated to form a new population. The new population is thereafter utilized in the algorithm's next iteration. Typically, the process ends after the population has attained a satisfactory fitness level or a maximum number of generations have been formed. A large number of possible solutions for a given problem can be formed by a GA, and then the evaluation of those solutions is performed for the decision of the fitness level of each solution.

**Fitness Value Calculation.** In GA, the value assigned to any individual of a population is termed the fitness value that is used to determine the probability of the optimal solution for an individual of the population. As much as the fitness value is higher, the solution will be better. Despite the fact that the GA cannot always identify the precise result, it always finds the optimal one. In this work, the fitness value of each individual is computed on the basis of three different criteria or NFRs, i.e., security, reliability, and usability.

**Initial Population Generation.** An adequate population size must be selected for the implementation of a GA so that the most favorable solution can be achieved. A small population size requires fewer amounts of computation time and effort while a big population

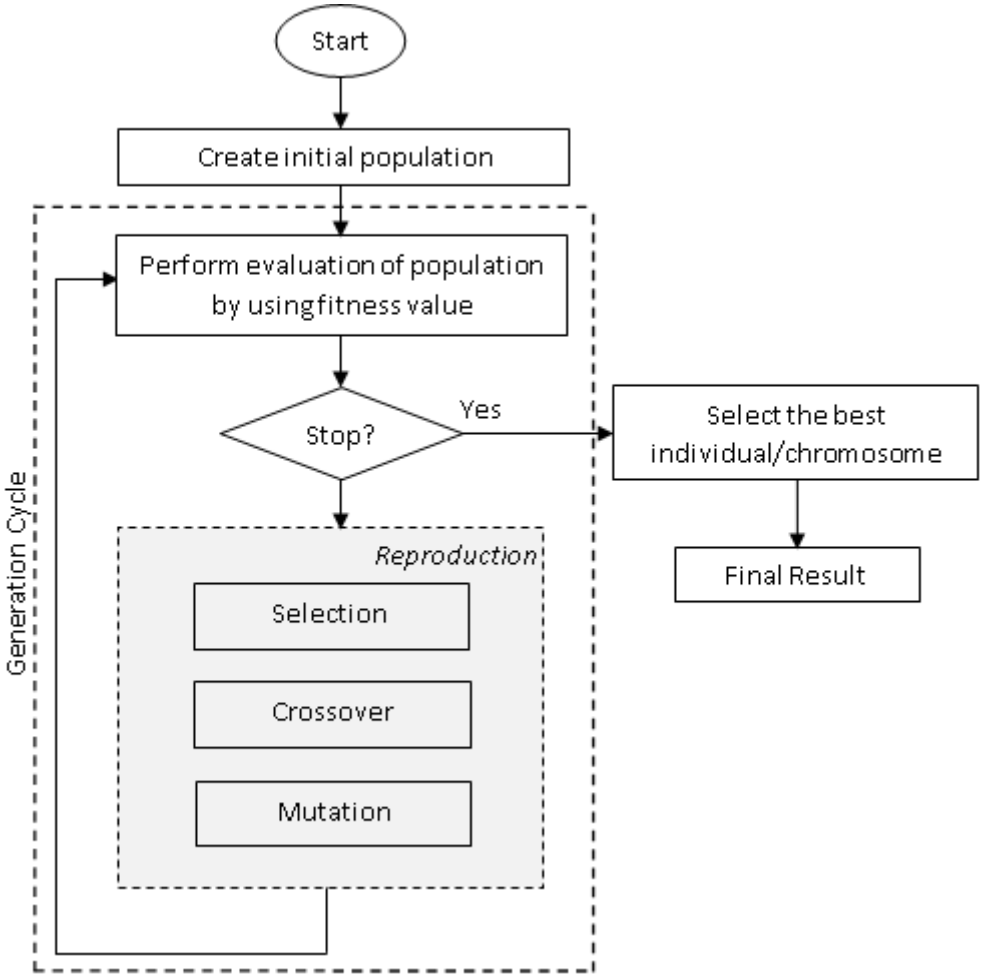


**Fig. 1.**The working of GA

size requires more, i.e., the computational time and the effort are directly proportional to the size of the population. We conducted five tests in our experiment by using populations of sizes 10,15,20,25, and 30. During the experiments, we kept the other factors constant. Four tests have been performed for each population size and then an average of the results obtained through all test runs is computed as illustrated in Table 1.

Despite the fact that there are 32 FRs available, the experimental results show that 20 is the ideal initial population size because the averages obtained for population sizes of 20 and 25 remain very stable, dropping just slightly from 83.50 to 82.75 as shown in Fig. 2.

**Process of Selection, Crossover, and Mutation.** After receiving inputs (i.e., criteria) from the user, a randomly initialized population is created for the evaluation of the fitness value. The result with the maximum fitness value is chosen and saved in a new population. Randomly two best parents are selected from the population by using the roulette selection

technique to perform the crossover operation. In roulette selection, the probability of a chromosome being selected is related to its fitness value. For example, let F1, and F2 are two different fitness variables having fitness values 25 and 21 respectively. If the defined fitness (DF) value is 20, then the roulette wheel will select F2 because (F2-DF) < (F1-DF). A new child chromosome is formed as a result of crossover operation. In our work, a single point crossover is used with a crossover rate of 0.5. The process of mutation is performed over the new child if its fitness value is less than the average fitness value; otherwise, the new child is added to the new population. The selection, crossover, and mutation processes are repeated up to n times of evolution where the fitness value remained constant.

**Crossover Rate, Mutation Rate, and Termination Criterion.** It's important to remember that the definition and interpretation of GA parameters like crossover rate, mutation rate, and the criterion for termination are all highly subjective and varied amongst systems. As a result, adequate experiments based on a population size of 20 have been carried out to determine the best values for these factors.

For crossover rates (0 to 1.0), the highest fitness values before and after the crossover for four tests are shown in Table 2. We also computed the difference among the highest fitness value before and after the processes of crossover. We analyzed that a crossover rate of 0.5 produces favorable outcomes. This is because the variations in the highest fitness values for the first three tests are positive, but the maximum fitness value for the fourth test remains unchanged. The variations in maximum fitness values before and after crossover are $\leq 0$ or nearby 0 for the remaining crossover rates, making them useless for application.

A mutation is a crucial component of a genetic algorithm because it keeps the population from being stuck at a locally optimal solution. It occurs as per the mutation rate set by a user. This rate should be minimal since a large mutation rate can transform the search into a rudimentary random search. We chose not to include a mutation rate in our work since mutation has the potential to turn a good solution into a bad one. However, the individual's fitness value will be calculated and compared to the average fitness value of the population. Any individual that has a fitness value lower than the average fitness value is mutated. This helps to conserve the population's good individuals and also allows them to regain valuable genetic material that may have been lost due to selection and crossover processes. To validate this, the tests are carried out with a mutation rate of 0.005 which is ~0. The highest and average fitness values computed for four tests before the mutation and after the mutation are illustrated in Table 3. The variations in higher and average fitness values before and after mutation are quite negligible, as shown in Table 3. This validates our use of a mutation probability of 0 in the GA while allowing individuals to undergo mutations selectively depending on fitness values.

**Table 1:** Calculation of initial population size

| Population Size | First Test | Second Test | Third Test | Fourth Test | Average |
|---|---|---|---|---|---|
| 10 | 42 | 47 | 40 | 43 | 43.00 |
| 15 | 63 | 72 | 51 | 60 | 61.50 |
| 20 | 81 | 84 | 90 | 79 | 83.50 |
| 25 | 94 | 78 | 71 | 88 | 82.75 |
| 30 | 91 | 89 | 93 | 87 | 90.00 |

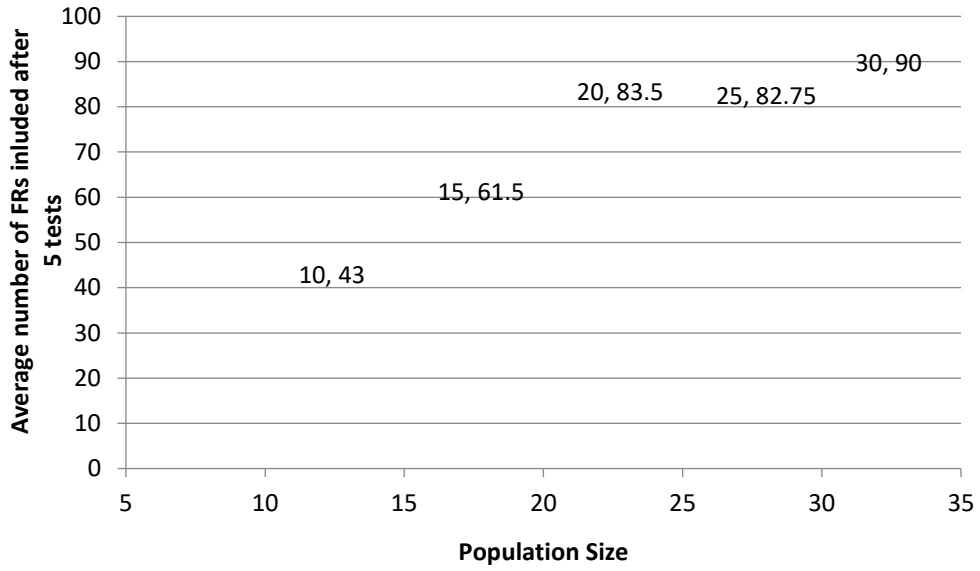


**Fig. 2.** Graphical representation of the experimental result

The GA is a probabilistic heuristic search. Obtaining an appropriate or ideal solution in the worst-case situation might take an indefinite time. So, to overcome this, a criterion should be specified to bring the search process to a halt after solutions have been determined. A predefined number of generations attained, the maximum fitness value achieved, and the time complexity metric are some of such criteria. For example, a time complexity metric, the maximum fitness value achieved, or a predefined number of generations attained are some of these criteria. The search procedure in our implementation ends after five generations, with the highest fitness values remaining constant after that, as presented in Table 4.

## 4 Experimental results and case study

The GA-based SRs selection function is implemented by using the Python programming language on the system with the configuration as Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz. Our implemented GA is capable to show its capacity to identify optimum solutions for various situations based on a preset set of criteria. As indicated in Table 5, four or five SRs with the highest fitness values are given for each situation to help in decision making for the selection of SRs.

To further verify the correctness of the solutions produced by GA, experiments have been carried out to acquire the average range of fitness values for five tests for a given situation. Table 6 shows that the range of fitness values for SRs selection under a given circumstance is close to each other. This consistency in results indicates that the system's best

**Table 2:** Effective crossover rate estimation

| Crossover Rate | Before crossover | After Crossover | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Maximum Fitness value | First test | | Second test | | Third test | | Fourth test | |
| | | Max | Difference | Max | Difference | Max | Difference | Max | Difference |
| 0.1 | 29.7601 | 29.7971 | 0.037 | 29.8771 | 0.117 | 29.8761 | 0.116 | 29.1321 | -0.628 |
| 0.2 | 29.7601 | 28.6213 | -1.1388 | 29.9012 | 0.1411 | 29.6651 | -0.095 | 30.0227 | 0.2626 |
| 0.3 | 29.7601 | 29.7052 | -0.0549 | 28.7502 | -1.0099 | 30.2871 | 0.527 | 29.7904 | 0.0303 |
| 0.4 | 29.7601 | 29.7601 | 0 | 29.2001 | -0.56 | 29.7702 | 0.0101 | 29.7601 | 0 |
| 0.5 | 29.7601 | 30.0114 | **0.2513** | 29.77 | **0.0099** | 30.9081 | **1.148** | 29.7601 | **0** |
| 0.6 | 29.7601 | 29.7601 | 0 | 29.7601 | 0 | 29.7901 | 0.030 | 30.3226 | 0.5625 |
| 0.7 | 29.7601 | 29.8626 | 0.1025 | 29.725 | -0.0351 | 27.0213 | -2.7388 | 28.9213 | -0.8388 |
| 0.8 | 29.7601 | 29.9215 | 0.1614 | 29.1901 | -0.57 | 29.2921 | -0.468 | 30.0921 | 0.332 |
| 0.9 | 29.7601 | 29.0004 | -0.7597 | 29.8982 | 0.1381 | 29.8603 | 0.1002 | 30.6654 | 0.9053 |
| 1.0 | 29.7601 | 29.788 | 0.0279 | 29.2644 | -0.4957 | 29.9001 | 0.14 | 29.7613 | 0.0012 |

**Table 3:** Variation in fitness values before and after mutation

| Mutation Rate at 0.005 | Before Mutation | After Mutation | | | |
|---|---|---|---|---|---|
| | | First Test | Second Test | Third Test | Fourth Test |
| Fitness Value (Max) | 29.7601 | 29.7601 | 30.0220 | 29.7924 | 29.9801 |
| Fitness Value (Average) | 24.9102 | 24.9102 | 25.0676 | 24.9929 | 25.0031 |

**Table 4:** Identification of termination condition

| Generation Number | Highest Fitness Value | | | |
|---|---|---|---|---|
| | After First Test | After Second Test | After Third Test | After Fourth Test |
| 1 | 30.0114 | 29.9012 | 30.9081 | 30.6654 |
| 2 | 30.0114 | 30.0010 | 31.2055 | 30.8962 |
| 3 | 31.7016 | 33.5508 | 31.6904 | 31.4278 |
| 4 | 31.8232 | 33.8927 | 32.0051 | 31.9998 |
| 5 | 31.7927 | **33.8229** | **32.5506** | **32.0004** |
| 6 | **31.9553** | **33.8229** | **32.5506** | **32.0004** |
| 7 | **31.9553** | **33.8229** | **32.5506** | **32.0004** |

**Table 5:** SRs selection based on different priorities

| **Case-1: Selecting SRs with 'Security' as the main priority** | | | | |
|---|---|---|---|---|
| **S.No.** | **Requirements** | **Security** | **Reliability** | **Usability** |
| 1 | Req3 | High | Medium | Low |
| 2 | Req7 | High | Medium | Medium |
| 3 | Req19 | High | Low | Low |
| 4 | Req23 | High | Medium | Low |
| **Case-2: Selecting SRs with 'Reliability' as the main priority** | | | | |
| 1 | Req5 | Low | High | Low |
| 2 | Req9 | Low | High | Medium |
| 3 | Req13 | Medium | High | Medium |
| 4 | Req22 | Low | High | Medium |
| 5 | Req27 | Medium | High | Medium |
| **Case-3: Selecting SRs with 'Usability' as the main priority** | | | | |
| 1 | Req1 | Medium | Medium | High |
| 2 | Req12 | Low | Low | High |
| 3 | Req16 | Medium | Low | High |
| 4 | Req25 | Medium | Low | High |
| 5 | Req28 | Medium | Low | High |

**Table 6:** The average fitness values comparison among different conditions

| Conditions | Security | Reliability | Usability | Average range of fitness values for 5 tests | Fitness value difference |
|---|---|---|---|---|---|
| 1 | Medium | High | High | 25.3351 – 25.4271 | 0.0920 |
| 2 | Low | High | High | 34.2709 – 34.3916 | 0.1207 |
| 3 | High | Medium | Medium | 26.2173 – 26.6272 | 0.4099 |
| 4 | High | Low | Medium | 27.2357 – 27.6051 | 0.3694 |
| 5 | Medium | High | Medium | 31.5603 – 31.8006 | 0.2403 |

solution is within a particular range. As a result, the system displays accuracy in selection based on the set of defined criteria.

The foregoing findings suggest that our proposed GA-based SRs selection approach is compatible with existing GA-based techniques and as well as the some other fuzzy based techniques like fuzzy AHP, fuzzy TOPSIS, GORS model, PRFGORE, etc. [18]. Yet, each study may still be improved; necessitating more research in some other aspects like computational complexity, validation of suggested techniques in other sectors, and assessment and selection of criteria.

## 5 Conclusion and future work

As shown in the case study, the GA can precisely produce optimal solutions for the SRs selection process. The suitable decision making for the selection of the requirements can lead to the success of a software product. The selection can be analyzed further by using subjective judgment. This improves supplier selection decision-making in a variety of contexts. There is, though, still scope for improvement. Some other criteria can also be considered to be used for the assessment of SRs. The search space and time complexity measurement can be used in future work to better understand and enhance the performance of the GA.

## References


[1] Sadiq, M., Devi, V. S.: Fuzzy-soft set approach for ranking the functional requirements of software. Expert Systems with Applications, Vol 193 (2021).

[2] Hamdia, K. M., Zhuang, X., Rabczuk, T.: An efficient optimization approach for designing machine learning models based on genetic algorithm. Neural Comput. & Applic., Vol. 33, pp. 1923-1933, 2021.

[3] Wu, C., Luo, W., Zhou, N., Xu, P., Zhu, T.: Genetic Algorithm with Multiple Fitness Functions for Generating Adversarial Examples. *2021 IEEE Congress on Evolutionary Computation (CEC)*. pp. 1792-1799 (2021).

[4] Nazim, M., Mohammad, C. W., Sadiq, M.: Analysis of fuzzy AHP and fuzzy TOPSIS methods for the prioritization of the software requirements. In: Kulkarni A.J. (eds) Multiple Criteria Decision Making. Vol. 407, pp. 79-90, Springer, Singapore (2022).

[5] Nazim, M., Mohammad, C. W., Sadiq, M.: Generating Datasets for Software Requirements Prioritization Research. In: IEEE International Conference on Computing Power, and Communication Technologies, Organized by Galgotias University, Greater Noida, India, pp. 1-6, (2020).

[1] Tan, F., Fu, X., Zhang, Y., Bourgeois, A. G.: A genetic algorithm-based method for feature subset selection," *Soft Comput.*, Vol. 12, pp. 111-120 (2008).

[2] Mosher, T.: Conceptual Spacecraft Design Using a Genetic Algorithm Trade Selection Process. Journal of Aircraft, Vol. 36, no. 1 (1999).

[3] Lin, C., Chen, H., Wu, Y.: Study of image retrieval and classification based on adaptive features using genetic algorithm feature selection. Expert Systems with Applications. Vol. 41, no. 15, pp. 6611-6621 (2014).

[4] Dorrington, K. P., Link, C. A.: Genetic-algorithm/neural-network approach to seismic attribute selection for well-log prediction. Geophysics, Vol. 69, no. 1, pp. 212–221 (2004).

[5] Golmohammadi, D., Creese, R. C., Valian, H., Kolassa, J.: Supplier Selection Based on a Neural Network Model Using Genetic Algorithm. In: IEEE Transactions on Neural Networks, Vol. 20, no. 9, pp. 1504-1519 (2009).

[6] Luan, J., Yao, Z., Zhao, F., Song, X.: A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization. Mathematics and Computers in Simulation, Vol. 156, pp. 294-309 (2019).

[7] Zhang, H., Cui, Y.: A model combining a Bayesian network with a modified genetic algorithm for green supplier selection. Simulation, Vol. 95, no. 12 (2019).

[8] Srivastava, P. R., Kim, T.: Application of Genetic Algorithm in Software Testing. International Journal of Software Engineering and Its Applications, Vol. 3 (2009).

[9] Finkelstein, A., Fuks, H.: Multiparty Specification. Proc. Fifth Int'l Workshop Software Specification and Design, pp.185-195 (1989).

[10] Kavitha, D., Ravikumar, S.: Software Security Requirement Engineering for Risk and Compliance Management. International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 10, no. 5, pp. 10-17 (2021).

[11] Xie, G., Li, Z., Yuan, N., Li, R., Li, K.: Toward effective reliability requirement assurance for automotive functional safety. ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 23, no. 5, pp. 1-26 (2018).

[12] Purnamasari, F., Hardi, S.: A Study on Usability Requirement for Redesigning Student Information System. 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM), pp. 145-148 (2019).

[13] Nazim, M., Mohammad, C. W., Sadiq, M.: Fuzzy based methods for the selection and prioritization of software requirements: A systematic literature review. In: 9th International Conference on Frontiers of Intelligent Computing: Theory and Applications, Springer, National Institute of Technology, Mizoram, India, pp. 1-14 (2021).

[14] Ahmad, J., Mohammad, C. W., Sadiq, M.: Identification of Security Requirements from the Selected Set of Requirements under Fuzzy Environment, 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp. 58-63 (2021).