International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 7 (2018) pp. 5167-5174

Test Case Retrieval Model by Genetic Algorithm with UML Diagram

Itti Hooda[(1)], Prof. Dr. Rajender Singh Chhillar[(2)]

[1]*Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak,(Haryana), India.*

[2]*Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak,(Haryana), India.*

Abstract

In this research paper we have shown the Reusability of test cases. If companies make repositories of Test cases then they can automatically extract the cases from Database, but the problem is matching of the test cases and the new Project UML Design. So we have used optimize weight model by Genetic algorithm which match the test cases by cosine similarity and extract the High Cosine similarity test cases which analysis on the basis of TPR and FPR (True positive rate and False positive rate). We can also find the accuracy of proposed process. We achieve significance high accuracy like in case of ATM 83%, Student management System 81.23% and Banking System 80.23%.

Keywords: FPR, IF Methods, LDA, BLUiR,UML, Mutation,Crossover,GA.

INTRODUCTION

Modern society depends on software-intensive systems. Software operates intangibly or very quietly in everything from kitchen appliances to critical infrastructure, and one's living a life exclusive of every day relying on systems running software needs a determined downshifting from life as most people enjoy it. As the importance of software constantly grows, so does the significance of being able to create it proficiently. Software development is an wide-ranging expression used to illustrate any approach to produce source code and its correlated documentation. Through the software crisis of the 1960s, it became clear that software difficulty quickly rises when software are scaled up to larger systems. The development techniques that were applied at the time did not result in needed software in a predictable manner. [7]

A powerful search engine that can retrieve software artifacts with high precision is obviously of great importance to developers. Software systems usually contain defects that need to be fixed after releases, and in some projects users are allowed to submit feedback on these defects that they encounter through bug reporting systems such as Bugzilla. Owing to this inherent complexity of software construction, software bugs remain frequent. For a large software system, the number of bugs may range from hundreds to thousands. However, performing this process manually for lot of bugs is time consuming and exclusive. Consequently, efficient techniques for locating bugs automatically from bug reports are highly desirable [1].In recent years, information retrieval (IR) based bug localization techniques have gained significant attention owing to their relatively low computational cost and minimal external dependencies. Effectively supporting software changes is essential to give a sustainable high-quality evolution of large-scale software systems, as realizing even a slight change may not be always straightforward. Software-change impact analysis, or basically impact analysis (IA), has been acknowledged as one such key protection activity. IA aims at estimating the potentially impacted entities of a system due to a proposed change [5].

The goal of contextual search is to incorporate a richer model of human searching behaviour into search systems, and as such, represents an opportunity for collaboration between information behaviour (IB) and information retrieval (IR), two fields which have hitherto progressed mostly in parallel. Researchers have worked towards maximizing the impact that software engineering research has on practice, for example, by providing methods and outputs that are as general (and thus as useful) as possible. A significant amount of research on applying Information Retrieval (IR) methods for analysing textual information in software artifacts has been conducted in the SE community in recent years. Software testing is indispensable for all software development. It is an integral part of the software engineering discipline. However, testing is labour-intensive and exclusive. It often accounts for more than 50% of total development costs. Thus, there are clear advantage in reducing the cost and improving the efficiency of software testing by automating the process. [9].

In fact, there has been a quick expansion of practices in using automated software testing tools. Currently, a large number of software test automation tools have been developed and have become accessible on the market. Between a range of testing activities, test case invention is one of the most intellectually demanding tasks and it is also of the most critical challenges, since it can have a tough impact on the effectiveness and efficiency of the whole testing process The goal of performance testing is to find performance problems, when an application under test (AUT) unexpectedly exhibits worsened characteristics for a specific workload. Existing feature location techniques use different tactics to find a feature's source code. Approaches based on information retrieval (IR) leverage the fact that identifiers knowledge to locate source code that is textually similar to a query describing a feature is there [4].

Various researchers have applied information retrieval methods to automatically search for appropriate files based on bug reports. They treat an initial bug report as a query and rank the source code files by their significance to the query. The developers can then analysis the returned files and fix the bug. These techniques are information retrieval based bug localization methods. Numerous of the obtainable IR-based

bug localization methods are anticipated in the context of feature/concept location, using a small number of selected bug reports for example, PROMESIR, etc[11].

LITERATURE REVIEW

[1] Describes a technology for automatically assembling large software libraries that promote software reuse by helping the user locate the components closest to his needs. [2] Propose a new approach leveraging information retrieval, in particular BM25-based document similarity function, to automatically predict the severity of bug reports. Moreover, investigated similar bug reports reported in the past and assign severity labels to newly reported bug reports. [3] build BLUiR on a proven, open source IR toolkit that anyone can use which provides a thorough grounding of IR-based bug localization research in fundamental IR theoretical and empirical knowledge and practice. And evaluated BLUiR on four open source projects with approximately 3,400 bugs. Results show that BLUiR matches or outperforms a current state-of-the-art tool across applications considered, even when BLUiR does not use bug similarity data used by the other tool.

[4] shows how version histories of a software project can be used to estimate a prior probability distribution for defect proneness associated with the files in a given version of the project. And showed that by just including the base models, the mean average precision (MAP) for bug localization improves by as much as 30%. [5] present an interactive-gaming-based teaching and learning platform called Pex4Fun. At the core of the platform is an automated grading engine based on symbolic execution. [6] Presents an adaptive approach to perform impact-analysis from a given change request to source code and conducted an empirical evaluation on four open source software systems. A benchmark consisting of a number of maintenance issues, such as feature requests and bug fixes, and their associated source code changes was established by manual examination of these systems and their change history.

[7] present research that combines information behaviour and information retrieval approaches to develop a contextual search system for a software engineering work domain. [8] described the techniques for automated test case, test data and test procedure generation for concurrent reactive real-time systems which are considered as the most important enablers for MBT in practice. With respect to the latter view, our experience with introducing MBT approaches in testing teams are sketched. [9] conducted a survey among traceability researchers and found that while a majority consider student artifacts to be only partly representative of industrial counterparts, such artifacts were typically not validated for industrial representativeness.

[10] combined ideas from representativeness and diversity and introduce a measure called sample coverage, defined as the percentage of projects in a population that are similar to the given sample and also introduced algorithms to compute the sample coverage for a given set of projects and to select the projects that increase the coverage the most.[11] builds new fundamental finding and proposes a novel solution to adapt, configure and effectively use a topic modelling technique, namely Latent Dirichlet Allocation (LDA), to achieve better (acceptable) and introduces a novel solution called LDA-GA, which uses Genetic Algorithms(GA) to determine a near-optimal configuration for LDA in the context of three different SE tasks: (1) traceability link recovery, (2) feature location, and (3) software artifact labelling.

[12] focusing on small and medium development for educational serious games and basing itself on two lines of research: agile development methodology and user-centered design (UCD) for children from 7 to 10 years. The agile methodology eXtreme Programming (XP) offers a useful option for the development of serious games as it establishes a continuous communication with all project stakeholders - including the end user - throughout the project, while UCD allows the user profile to be known and identified so that the game will meet the needs and match the capabilities, expectations and motivations of the child.

[13] the paper aims at giving an introductory, up-to-date and (relatively) short overview of research in automatic test case generation, while ensuring a comprehensive and authoritative treatment. Also presents an orchestrated survey of the most prominent techniques for automatic generation of software test cases, reviewed in self-standing sections.

[14] offering a novel solution for finding performance problems in applications automatically using black-box software testing. implemented our solution and applied it to a medium-size application at a major insurance company and to an open-source application. Performance problems were found automatically and confirmed by experienced testers and developers.

[15] describe the Sourcerer infrastructure, present the applications that they have built on top of it, and discusses how existing tools could benefit from using Sourcerer .have developed Sourcerer, an infrastructure for large-scale collection and analysis of open source code. [16] applies the idea of data fusion to feature location, the process of identifying the source code that implements specific functionality in software. A novel contribution of the proposed model is the use of advanced web mining algorithms to analyze execution information during feature location. [17] build upon a prior body of work to highlight the state-of-the-art in software traceability, and to present compelling areas of researchthat need to be addressed.

[18] have developed REST-bench on the sound fundamentals of a taxonomy on REST alignment methods and validated the method in five case studies. [19] propose Bug Locator, an information retrieval based method for locating the relevant files for fixing a bug. BugLocator ranks all files based on the textual similarity between the initial bug report and the source code using arevised Vector Space Model (rVSM).

[20] proposed a recommender (called Refoqus) based on machine learning, which is trained with a sample of queries and relevant results and evaluated Refocus empirically against four baseline approaches that are used in natural language document retrieval.

[21] To identify, evaluate and synthesize research published by software engineering researchers concerning their experiences of performing SRs and their proposals for

improving the SR process.[22] analysed reported patches for three existing generate-and-validate patch generation systems (GenProg, RSRepair, andAE) and also present Kali, a generate-and-validate patch generation system that only deletes functionality.[23] analysing the version history, such tangled changes will make all changes to all modules appear related, possibly compromising the resulting analyses through noise and bias.

[24] described an innovative tool called GUITAR that supports a wide variety of GUI testing techniques and demonstrate these features of GUITAR via several carefully crafted case studies. [25] proposed an automated technique, called SwiftHand, for generating sequences of test inputs for Android apps, main goal is to simply guide test execution into unexplored parts of the state space.[26] explores some of the relationships between these strands of closely related work, arguing that they have much in common and sets out some future challenges in the area of AI for SE.

[27]presents FAMILIAR a Domain-Specific Language (DSL) that is dedicated to the large scale management of feature models and that complements existing tool support. They illustrated how an SPL consisting of medical imaging services can be practically managed using reusable FAMILIAR scripts that implement reasoning mechanisms.

[28] addresses the problem of concept location in source code by proposing an approach that combines Formal Concept Analysis and Information Retrieval and proposed approach, Latent Semantic Indexing, an advanced Information Retrieval approach, is used to map textual descriptions of software features or bug reports to relevant parts of the source code, presented as a ranked list of source code elements.

[29] presents a novel EFSM inference technique that addresses the problems of inflexibility and non-determinism. It also adapts an experimental technique from the field of Machine Learning to evaluate EFSM inference techniques, and applies it to three diverse software systems.

PROPOSED METHODOLOGY

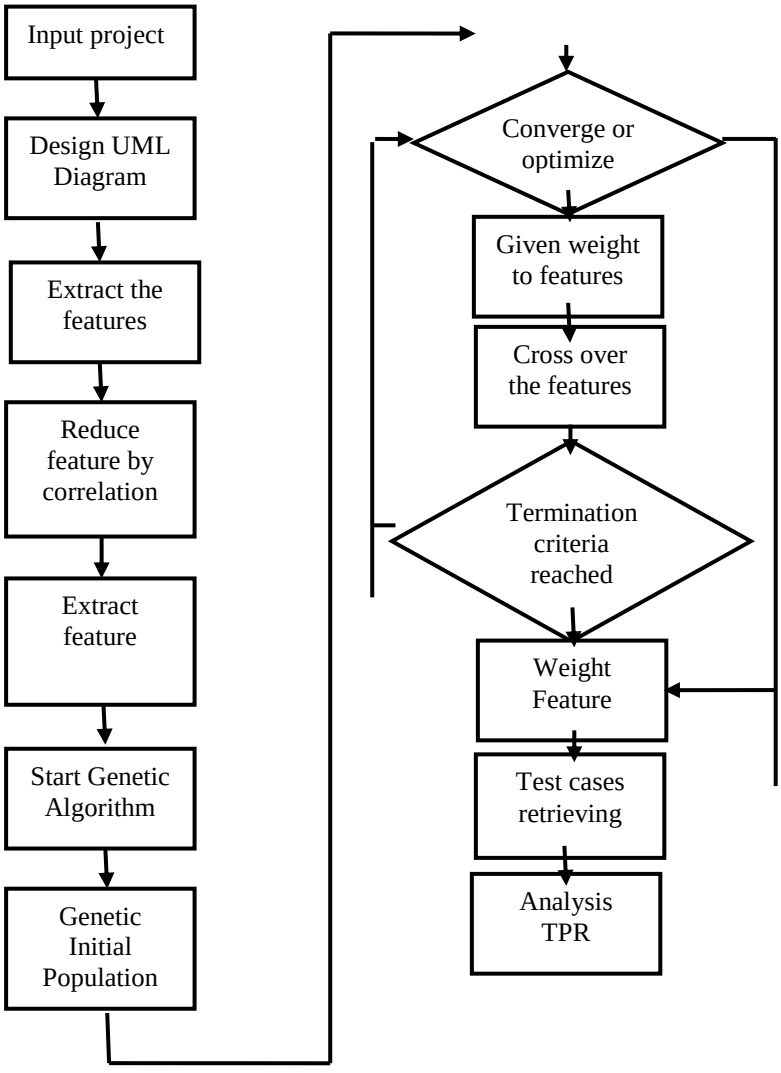


Figure 1. Block diagram of research methodology.

ALGORITHM

Input:

The UML Diagram to be tested U;
The flow of graph (CFG) and attributer (ADG) of U;
The population size (PS)
Maximum number of Generation (MG);
Probability of crossover (PX);
Probability of mutation (PM);

Output:

TRP of Test Case Retrieval

Begin

Step 1: Initialization

for i= 1 to PS
Initialize each feature ⟵ U_w ϕ;
Initialize the weight ⟵ U_w ϕ;
nRum ⟵ 0;

Step 2: Retrieval test cases

While (Test cases not finish)
Begin
nRum ⟵ nRum +1
for i=1 to PS
Put each U_i ⟵ {w_o.w_k}
Current_Population ⟵ Initial_Poplulation
No_Of_Generation ⟵ 0;
For each individual of current population do
Begin
Convert current chromosome according to features;
If (current features is optimize) then
nfeatures ⟵ nfeatures +1
End If
End For;
While (the Best Individual is not Independent feature and No_of_Generations ← MG)
Begin
Select set of parents of new population from members of current population using roulette wheel method;
Generate New_population using crossover and mutation operations;
Current_Population ⟵ New_Population;
For each Individual of Current_Population do
Begin
Given the weight of features
nfeatures ⟵ nfeatures + 1
End
No_Of_Generation ⟵ No_Of_Generation +1
While (Test Case)
Weighted features match with use case
Database
Select Test Cases;
Analysis TRP
End while

GENETIC ALGORITHM FOR FEATURE SELECTION

In stochastic heuristic optimization methods, genetic algorithm belongs. The multi- dimensional optimization problem solving is the main usage of GAs where there is no known analytic solution or use is not possible.

An optimization problem is denoted as

$$\bar{x}_{opt} = \arg\ opt\{f(\bar{x}) \mid \bar{x} \in C\} \quad (1)$$

where

f is the cost function,

C is the set of feasible solutions,

$\bar{x}$ is a feasible solution and

$\bar{x}_{opt}$is the optimum weight to features.

In this strategy the set C used for searching is very important. To generate all feasible solutions 'step by step' is the most primitive strategy, calculate the cost function for every solution and to find the best solution. This procedure in practice cannot be used. An infinite set is theoretically C. When computer is used, the C is finite, because in memory, infinite number of combination is stored. The set C having number of different elements is always very high. So, all possible values of $\bar{x}$ cannot be explored in a suitable time. Some kind of more clever strategy is used by Heuristic methods.

In this strategy of searching, GA uses the set C is inspired with natural evolution, where to survive best weight have biggest chance and new offspring is to become parents. Additionally, in the nature – mutation, another existing mechanism is used by GA.

A 'small' random change of genetic information is mutation. The living conditions changes are easier to adapt by using mutation in features weight. Degeneration can also be prevented by using mutation (the deadlock for optimization in a local extreme is supposed to be an analogy to degeneration).

GA works with the whole population of solutions instead of just one solution in time and has an iterative character. Gas are simplified version and by using computer only they can be solved. The reason can be seen here, why GA formal description is neither unified nor commonly accepted. To support future implementation some formal description is needed. The operators and parameters contained by stochastic heuristic algorithm that is GA, is as follows:

$$GA = (N, P, f, \Theta, \Omega, \Psi, \tau) \quad (2)$$

where P is the population containing N elements (features)

$P = \{S1, S2, \ldots, SN\}$.

Each element Si, $i = 1, \ldots, N$ represents the solution of the problem is a string (or set) of integers of the fix length of n, so $Si \in Zn$. f denotes the cost function (fitness function) in which every individual is assigned with real number:

$$f: Si \rightarrow R,\ i = 1, \ldots, N. \quad (3)$$

Θ is the parent selection operator, which u element is selected from P :

$$\Theta: P \rightarrow \{S1, S2, \ldots, Su\}. \quad (4)$$

Where Ω is the set of genetic operators $\Omega*$containing the crossover operator Ωc,

Ωm is the mutation operator.

v descendants generated by all these operators $\{O1, O2, \ldots, Ov\}$ from u parents $\{S1, S2, \ldots, Su\}$:

$$\Omega*: \{S1, \ldots, Su\} \rightarrow \{O1, \ldots, Ov\}. \quad (5)$$

Ψ is a deletion operator, and by a set of population elements, v selected elements is deleted from the population in the t-th iteration $P(t)$modeled. By adding (5) to the population, new v children is generated and created a new population $P(t + 1)$:

$$P(t + 1) = P(t) - \Psi(P(t)) \cup \{O1, \ldots, Ov\} \quad .(6)$$

τ is the stopping rule :

$$\tau(P(t)) \rightarrow (\text{true, false}). \quad (7)$$

One or more chromosomes contained in feature Si dependent on its implementation. In the computer system the inner representation of the chromosome is the chromosome genotype and is treated as a vector of genes.

An abstract mathematical object representing by chromosome phenotype $\bar{x}$ in solution of solved optimization problem. Numbers, symbols, the graph structure description etc containing these treated as vector.

In the memory of the chromosome is stored in the field bits form and expected is the use of computer, finite length should be in this field. Means chromosomes set of all possible phenotypes is also finite:

$$X = \{\bar{x}, \bar{x}', \bar{x}'', \ldots\ldots\}$$

When only one chromosome contained in the element of population Si and the cost function (3) can be written in the following form:

$$f: X \rightarrow R.$$

Then the original problem (1) solution may be transformed the cost function f can be searched for global minimum:

$$\bar{x}_{opt} = \arg \min_{\bar{x} \in X} f(\bar{x}) \quad (8)$$

To create a chromosome is the ambition of evolution, in which optimal chromosome $\bar{x}_{opt}$ is very near (or even equal). Where pm is denoted as the probability of mutation and pc as the probability ofcrossover. The chromosome must satisfy therule to the application of the mutation operator :

$$\lim_{pm \rightarrow 0} \Omega_m(\{\bar{x}\}) = \{\bar{x}\} \quad .(9)$$

The probability of selection is determined by probability of crossover pc, the specific individual (chromosome) for reproduction process. The value of pc depends on the fitness and is different for every individual. The fitness F is a non-negative number and in the current population bringing a relative quality measure of every individual. The mapping function

$$F: P \rightarrow R^+$$

the rule needs to be satified

$$f(\vec{x}_1) \le f(\vec{x}_2) \Rightarrow F(\vec{x}_1) \ge F(\vec{x}_2)\,. \quad (10)$$

Denoting the relative (standardized) fitness as

$$F'(\vec{x}) = \frac{F(\vec{x})}{\sum_{\vec{x} \in P} F(\vec{x})} \quad (11)$$

where

$$\forall \vec{x} \in P : 0 \le F'(\vec{x}) \le 1\,, \quad (12a)$$

$$\sum_{\vec{x} \in P} F(\vec{x}) = 1 \quad (12b)$$

The value of probability of crossover p_c directly used frequency is the relative fitness:

$$P_c(\vec{x}) = F'(\vec{x})\,. \quad (13)$$

Using following sequence the run of GA can be described:

1. Usually random generation is used for the generation of initial population.
2. All features fitness are computed.
3. (4) for parent selection and (5) for offspring generation.
4. By using deletion operator creating new population and in step (6) generated offspring is added.
5. Then mutation takes place.
6. Get back to step 3, in case stopping rule is not satisfied.
7. In the population, the result is the best individual.

RESULT AND ANALYSIS

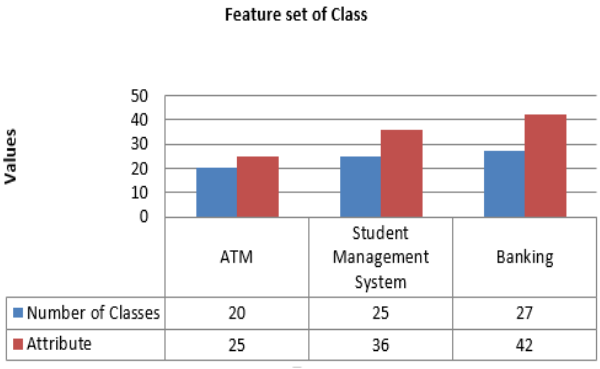


Figure 2. Feature set of Class

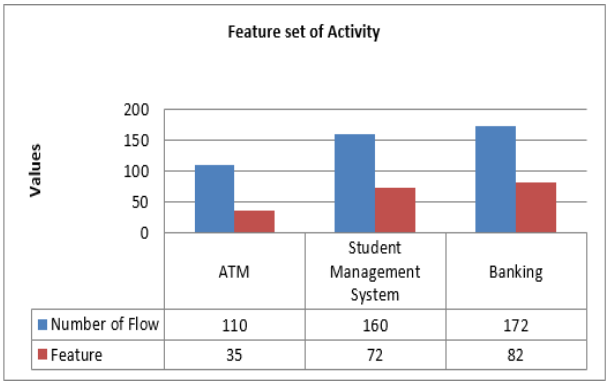


Figure 3. Features Set of Activity.

GENERIC ALGORITHM WITHOUT FEATURES SELECTION

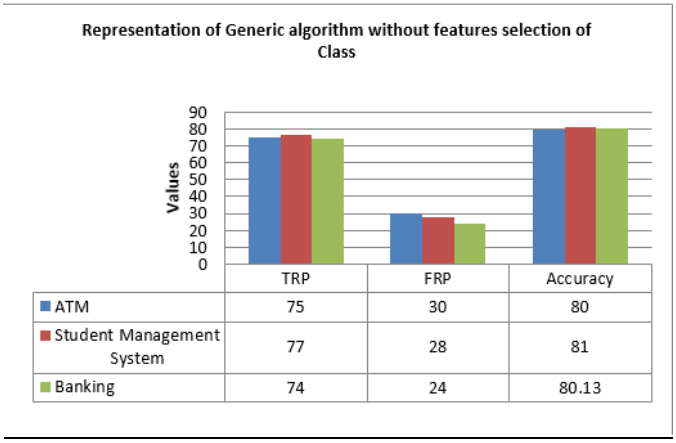


Figure 4. Representation of Generic algorithm without features selection of class.

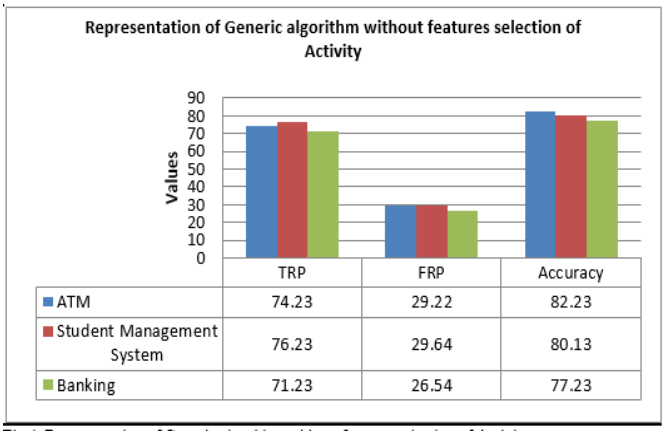


Figure 5. Representation of Generic algorithm without features selection of Activity.

GENERIC ALGORITHMS WITH CORRELATIONS :

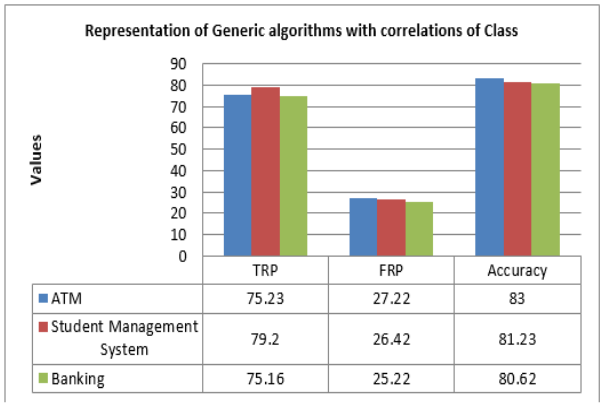
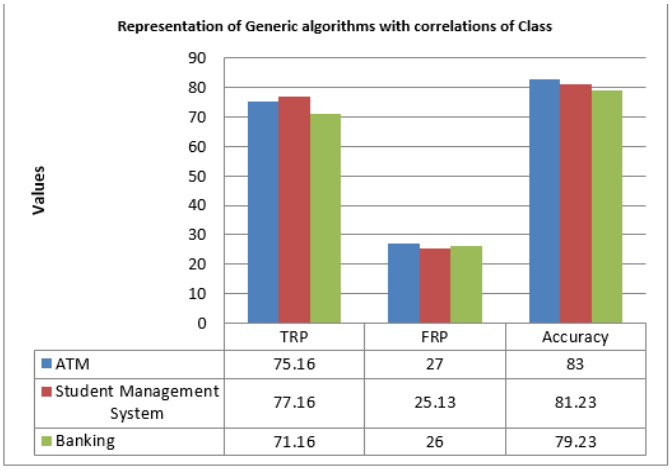


Figure 6. Representation of Generic algorithms with correlations of Class.



	TRP	FRP	Accuracy
ATM	75.16	27	83
Student Management System	77.16	25.13	81.23
Banking	71.16	26	79.23

Figure 7. Representation of Generic algorithms with correlations of Class

CONCLUSION

In this paper we have shown the test cases Retrieval by Genetic Algorithm optimization and matching process. But the major challenge was which design and feature should be used for that. So we used Object oriented UML designing method. Features are also reduced by correlation method. In this paper we have compared two UML Design class diagram and activity Diagram with and without correlation method. We conclude with the result that correlation show more accuracy than without correlation. In future we can enhance our work with more cases and optimize by learning and optimization method.

REFERENCES

[1] Maarek, Y. S., Berry, D. M., & Kaiser, G. E.,"An information retrieval approach for automatically constructing software libraries", IEEE Transactions on software engineering, Vol. 17, No. 8, 1991, pp. 800-813.

[2] Tian, Y., Lo, D., & Sun, C.,"Information retrieval based nearest neighbor classification for fine-grained bug severity prediction", 19th Working Conference on Reverse Engineering, IEEE, October 2012, pp. 215-224.

[3] Saha, R. K., Lease, M., Khurshid, S., & Perry, D. E.,"Improving bug localization using structured information retrieval",Automated Software Engineering (ASE), IEEE/ACM 28th International Conference, November 2013, pp. 345-355.

[4] Sisman, B., &Kak, A. C.,"Incorporating version histories in information retrieval based bug localization", Proceedings of the 9th IEEE Working Conference on Mining Software Repositories June 2012, pp. 50-59.

[5] Tillmann, N., De Halleux, J., Xie, T., Gulwani, S., & Bishop, J.,"Teaching and learning programming and software engineering via interactive gaming", 35th International Conference on Software Engineering (ICSE),IEEE, May 2013, pp. 1117-1126.

[6] Gethers, M., Dit, B., Kagdi, H., &Poshyvanyk, D.,"Integrated impact analysis for managing software changes",34th International Conference on Software Engineering (ICSE),IEEE, June 2012, pp. 430-440.

[7] Freund, L., & Toms, E. G.,"Contextual search: from information behaviour to information retrieval", Proceedings of the Annual Conference of CAIS/Actes du congrèsannuel de l'ACSI.

[8] Peleska, J.,"Industrial-strength model-based testing-state of the art and current challenges", arXiv preprint arXiv:1303.1006, 2013...

[9] Borg, M.,"Advancing Trace Recovery Evaluation-Applied Information Retrieval in a Software Engineering Context", arXiv preprint arXiv:1602.07633, 2016.

[10] Nagappan, M., Zimmermann, T., & Bird, C.,"Diversity in software engineering research", Proceedings of the 2013 9th joint meeting on foundations of software engineering,ACM, 2013, August pp. 466-476.

[11] Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshyvanyk, D., & De Lucia, A.,"How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms",Proceedings of the 2013 International Conference on Software Engineering, IEEE, 2013, May, pp. 522-531.

[12] Cano, S. P., González, C. S., Collazos, C. A., Arteaga, J. M., & Zapata, S., "Agile Software Development Process Applied to the Serious Games Development for Children from 7 to 10 Years Old", International Journal of Information Technologies and Systems Approach (IJITSA), 2015, Vol. 8, No. 2, pp. 64-79.

[13] Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., ...& McMinn, P.,"An orchestrated survey of methodologies for automated software test case generation", Journal of Systems and Software, Vol. 86, No. 8, 2013, pp. 1978-2001.

[14] Grechanik, M., Fu, C., &Xie, Q.,"Automatically finding performance problems with feedback-directed learning software testing", 34th International Conference on Software Engineering (ICSE), IEEE, 2012, June, pp. 156-166.

[15] Bajracharya, S., Ossher, J., & Lopes, C.,"Sourcerer: An infrastructure for large-scale collection and analysis of open-source code", Science of Computer Programming, 2014, Vol. 79, pp. 241-259.

[16] Dit, B., Revelle, M., &Poshyvanyk, D.,"Integrating information retrieval, execution and link analysis algorithms to improve feature location in software", Empirical Software Engineering, 2014, Vol. 18, No. 2, pp. 277-309.

[17] Cleland-Huang, J., Gotel, O. C., Huffman Hayes, J., Mäder, P., &Zisman, A.,"Software traceability: trends and future directions",Proceedings of the on Future of Software Engineering, ACM, 2014, May, pp. 55-69.

[18] Unterkalmsteiner, M., Gorschek, T., Feldt, R., &Klotins, E., "Assessing requirements engineering and software test alignment—Five case studies", Journal of Systems and Software, Vol. 109, pp. 62-77.

[19] Jian, Z., ZHANG, H., & David, L.,"Where should the Bugs be Fixed? More Accurate Information Retrieval-Based Bug Localization based on Bug Reports",34th International Conference on Software Engineering (ICSE 2012), pp. 14-24.

[20] Haiduc, S., Bavota, G., Marcus, A., Oliveto, R., De Lucia, A., & Menzies, T.,"Automatic query reformulations for text retrieval in software engineering", Proceedings of the 2013 International Conference on Software Engineering, IEEE, 2013, May, pp. 842-851.

[21] Kitchenham, B., & Brereton, P.,"A systematic review of systematic review process research in software engineering", Information and software technology, 2013, Vol. 55, No. 12, pp. 2049-2075.

[22] Qi, Z., Long, F., Achour, S., &Rinard, M.,'An analysis of patch plausibility and correctness for generate-and-validate patch generation systems",Proceedings of the 2015 International Symposium on Software Testing and Analysis, ACM, 2015, July, pp. 24-36.

[23] Herzig, K., Just, S., & Zeller, A, "The impact of tangled code changes on defect prediction models", Empirical Software Engineering, 2016, Vol. 21, No. 2, pp. 303-336.

[24] Nguyen, B. N., Robbins, B., Banerjee, I., &Memon, A.,"GUITAR: an innovative tool for automated testing of GUI-driven software", Automated Software Engineering, 2014, Vol. 21, No. 1, pp.65-105.

[25] Choi, W., Necula, G., & Sen, K.,"Guided gui testing of android apps with minimal restart and approximate learning", ACM SIGPLAN Notices, ACM, 2013, October, Vol. 48, No. 10, pp. 623-640.

[26] Harman, M., "The role of artificial intelligence in software engineering", Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering, IEEE, 2012, June, pp. 1-6.

[27] Acher, M., Collet, P., Lahire, P., & France, R. B., "Familiar: A domain-specific language for large scale management of feature models", Science of Computer Programming, 2013, Vol. 78, No. 6, pp. 657-681.

[28] Poshyvanyk, D., Gethers, M., & Marcus, A.,"Concept location using formal concept analysis and information retrieval", ACM Transactions on Software Engineering and Methodology (TOSEM), 2014, Vol. 21, No. 4, pp. 23.

[29] Walkinshaw, N., Taylor, R., & Derrick, J.,"Inferring extended finite state machine models from software executions", Empirical Software Engineering, 2015, Vol. 21, No. 3, pp. 811-853.

[30] Wieringa, R. J., "Design science methodology for information systems and software engineering", Springer, 2014.