Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 6, No. 1, March 2018, pp. 105~109
ISSN: 2089-3272, DOI: 10.11591/ijeei.v6i1.395

Dynamic Integrated System for Detecting and Fixing Vulnerability Bugs

R. Anitha*[1], M.V.Srinath[2]
[1]Department of Computer Science, Sengamala Thayaar Educational Trust Women's College, Sundarakkottai, Mannargudi, Tamilnadu, India
[2]Sengamala Thayaar Educational Trust Women's College, Sundarakkottai, Mannargudi, Tamilnadu, India

Article Info

Article history:

Received Jan 9, 2018
Revised Jan 27, 2018
Accepted Feb 7, 2018

Keyword:

Bug rejection
Clustering data
EM
Software quality
Vulnerability testing

ABSTRACT

Bugs are one of the important barriers in the field of software development. Concurrent and frequent bugs are non-deterministic in nature and in the time of vulnerability testing. It is difficult to detect the dynamic bugs with a high representation of vulnerability that causes the damage to the software products. Existing vulnerability testing methods relied on the conventional static testing techniques of finding and fixing the bugs but it does not show a high ratio of they handle or require specific hardware support. It does not include in the clustering approach. To overcome the limitations in the existing techniques, this proposed methods Modified Particle Swarm Optimization (MPSO), Expectation Maximization (EM) Clustering and Variable Neighborhood search. The primary input dataset is preprocessed to obtain the relevant and irrelevant data partition and optimized dataset was given as input to the Modified Particle Swarm Optimization (MPSO) technique

Corresponding Author:

R. Anitha,
Department of Computer Science,
Sengamala Thayaar Educational Trust Women's College,
Sundarakkottai, Mannargudi, Tamilnadu, India.
Email: veeanitha@yahoo.in

1. INTRODUCTION

Software Quality Assurance is used to ensure the quality of software in the field of software product development. It will improve the standard of the out coming Software Bugs are one of the important factors in STLC (Software Testing Life Cycle). The increase of bugs will reduce the software quality. So bug detection or prediction will be helpful to the software developers and testers. It also includes the number of bugs, non-trivial bugs, number of major bugs, number of critical bugs, number of high priority bugs. Using the information the vulnerable part of the software can be identified. The identification will improve the software quality assurance. The fundamental idea is to gather insights portraying a program's runtime conduct over numerous executions. Clustering analysis concentrates on purely numerical data only. The typical clustering algorithms include the k-means, EM algorithm, and their variants. The bug rejection using to software testing maintained. The data mining is used to the clustering technologies. They result in two algorithm execution on the software quality assurances testing and vulnerability detection on the work.

2. PROPOSED METHOD

The proposed methodology was the combination of three algorithms such as Modified Particle Swarm Optimization (MPSO), Expectation Maximization (EM) Clustering and Variable Neighborhood search. The input dataset is preprocessed to prepare the datasets to cluster the irrelevant data removed from

the dataset, then the entropy values are calculated to get the maximum vulnerability level of test cases. The preprocessed dataset is split up into several subsets. Feature selection is used as next step to reduce the execution time of the data. Modified Particle Swarm Optimization (MPSO) is used to predict the bugs in eeffective way. Feature reduced data is split up into several clusters. For clustering Expectation Maximization (EM) clustering is used to cluster the bugs in repeated manner. Variable Neighborhood Search algorithm is used to find the sensitive data in the dataset. This sensitive data will make the software as more vulnerable. Vulnerability detection will definitely improve the software quality assurance.

ALGORITHM 1 MPSO

Preliminaries and Assumptions

FEATURE SELECTION (MODIFIED PARTICLE SWARM OPTIMIZATION)

E ← *entropy*

Step1:for*i=0*: *A***do**

$E_i = \sum_{j=1}^{Nod} -p_j log_2 p_j$

Step 3:end for

Step 4:for*j=0*: *A.size-1***do**

Step 5: $E_j = \sum_{k=1}^{Nod.j} P_k E_k$

Step 6:end for

Step 7:Ig ← *information gain*

Step 8:Ba ← *best attributes*

Step 9:for*j=0*: *A.size-1***do**

Step 10: $Ig = E_t - E_j$

if $Ig_j > max$ **then**

$max = Ig_j$

$Ba = Ig_j$

Step 11:end if

Step 12: end for

The MPSO performs the head clustering and calculate the entropy values from the input datasets that is equally proportional to the sum of input vulnerability test case datasets. It include the size of values in entropy detection found in the first method, then the feature selection can be processed in the next step by using the EM technique.

Expectation Maximization Clustering

Noc ←*number of cluster*

Chs ←*cluster heads*

T←*total*

Hp ←*head probability*

for*i=0*: *Noc***do**

$Chs_i = select\ cluster\ head$

for*j=0*: *A***do**

$T += Chs_{ij}$

end for

$Hp_i = T/A.Size$

end for

Np ←*node probability*

Ed ←*euclidean distance*

M ←*maximization*

Sh ←*selected head*

for*i=0*: *Nod***do**

for*j=0*: *A***do**

$T += Chs_{ij}$

end for

$Np_i = T/A.Size$

for*j=0*: *Noc***do**

$Ed_{ij} = \sqrt{\sum_{k=1}^{A} (Nod_{ijk} - Chs_{ijk})^2}$

$$M = (Hp)^{Np} * (1 - Hp)^{Np} * Ed_{ij}$$

if $M > max$ **then**

$$max = M$$
$$Sh = Noc_j$$

end if

end for

end for

The number of holo-entropy and entropy of each vulnerability cluster heads was given as input into the EM to reduce the dynamic bugs by doing the iteration process with constrained threshold value, then it chooses the cluster data size and data clustering in maximum values. The maximum values of vulnerability test cases moves into the maximum queue.

Step 4: Vulnerability filtering in dataset in maximum queue.
Variable Neighborhood Search
$T \leftarrow temp$
for $i=0$: Nod **do**
for $j=0$: Nod **do**

$$Ed_{ij} = \sqrt{\sum_{k=1}^{A} \left(Nod_{ijk} - Chs_{ijk}\right)^2}$$

if $Ed > min$ **then**

$$min = Ed$$
$$T = Nod_j$$

end if

end for

end for

Search neighborhood vulnerability a value to maximum vulnerability test cases with threshold vulnerability value used by the system and returns the maximum and data minimum values to predict to the final daataset.

3. RESULT AND DISCUSSION

The software quality assurance checking on MPSO and EM algorithm is designed for reliable and effectivw bg predictions. It had been used to a number of clustering data. Figure 1 expresses the bug detection details with filteration reports.

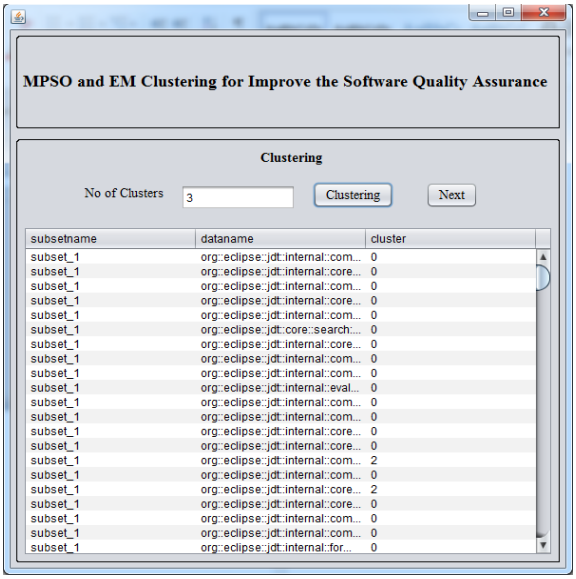

Figure 1. Bugs data filteration

This result is testing with vulnerability detection from a cluster in software quality assurances. Bug-free software is having high quality. It also includes the number of bugs, non-trivial bugs, number of major bugs, number of critical bugs, number of high priority bugs in Figure 2. Using this information the vulnerable part of the software can be identified. This identification will improve the software quality assurance. The fundamental idea is to gather insights portraying a program's runtime conduct over numerous executions.

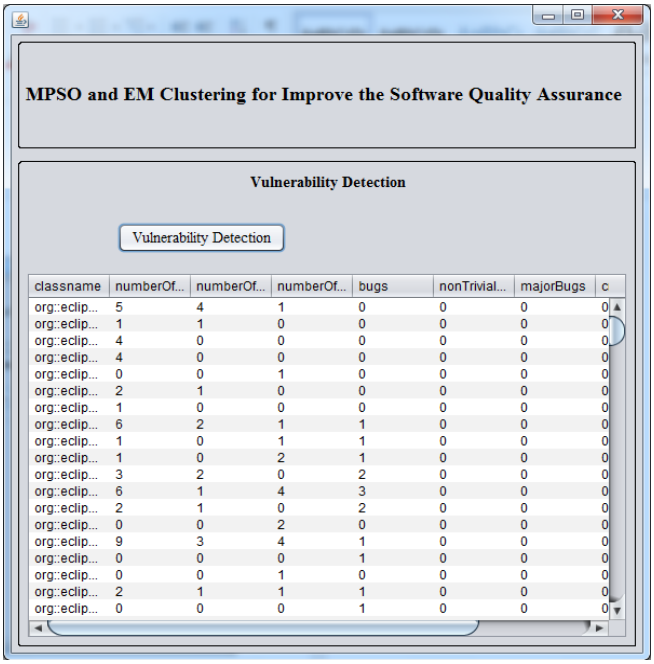

Figure 2. Bug detection reports

The method used Modified Particle Swarm Optimization (MPSO), Expectation Maximization (EM) Clustering and Variable Neighborhood search for vulnerability test cases. At preprocessing irrelevant data are removed from the dataset and moved into the EM mechanism. Then preprocessed dataset was split up into several subsets. Therein data cluster to the process will be done by the EM and vulnerability filter was increased drastically by this proposed work whiose details are explained in Figure 3. The quality assurances in volatility detection in the execution of this process had given the maximum output ranges when comparing to existing techniques.

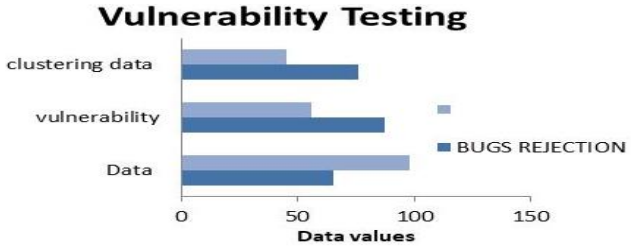

Figure 3. Vulnerability testing results using proposed method

The process was improved to vulnerability detection because of three step process by this proposed method includes clustering of each test case heads and then entropy and holo-entropy values were moved to found the software quality check test in bug error correction .the chart explain to the improvement to the high-level process. First software data upload to the bug and clustering data then software quality assurances data detection.

Finally, many solutions have been introduced to help fixing the dynamic vulnerability bugs. These all three dynamic methods that integrate dynamic bug detection and fixing comparing with static methods that generate that shows bugs again offline. This model have been simulated using Java 8 with dynamic test cases with different parameters shown the better bug detection and fixing and also controls both atomicity violations and order violations.

4. CONCLUSION

The proposed technique was shown the success rates in vulnerability bug rejection in software quality assurance environments by applying different of testing parameters. The process was tested with threshold values and vulnerability constraints at each stage. The combination of three algorithms is integrated such as Modified Particle Swarm Optimization (MPSO), Expectation Maximization (EM) Clustered and Variable Neighborhood search. It was preprocessing the irrelevant data and vulnerability test case validation. Then preprocessed dataset is split up into several subsets. Feature selection is used as next step to reduce the execution time of the data. The vulnerability test was detected with different parameters set and achieved the maximum bug reduction, that shows this proposed method for vulnerability testing experiments was executed in the software quality assurances test beds. In future this system will be tested in software product based vulnerability environments.

REFERENCES

[1] Liu, Wangshu, Shulong Liu, Qing Gu, Jiaqiang Chen, Xiang Chen, and Daoxu Chen, "Empirical Studies of a Two-Stage Data Preprocessing Approach for Software Fault Prediction," *IEEE Transactions on Reliability*, vol. 65, no. 1, pp38-53,2016.

[2] Jafar Al-Kofahi, and Lisong Guo*"Static detection of configuration-dependent bugs in configurable software"* , Proceeding of ICSE '15 Proceedings of the 37th International Conference on Software Engineering, vol.2,pp 795-796,2015.

[3] Arun Reungsinkonkarn*,"Bug Detection Using Particle Swarm Optimization with Search Space Reduction"* In 2015 6th IEEE International Conference on Intelligent Systems, Modelling and Simulation (ISMS),pp. 53-57, 2015

[4] Sokratis Tsakiltsidis*, "On Automatic Detection of Performance Bugs",* IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp.132-139, 2016.

[5] B. Caglayan, A. T. Misirli, A. B. Bener, and A. Miranskyy, "Predicting defective modules in different test phases," *Software Quality Journal*, vol. 23, no. 2, pp. 205–227, 2015

[6] B. Lucia, L. Ceze, and K. Strauss, *"Color Safe: Architectural sup- port for debug and with passion avoiding multi-variable atomicity violation,"* In Proc. 37th Annu. Int. Symp. Comput. Archit., 2010, pp. 222–233.

[7] E. Weyuker, T. Ostrand, and R. Bell, "Comparing the effectiveness of several modeling methods for fault prediction," *Empirical Softw. Eng*., vol. 15, no. 3, pp. 277–295, 2010.

[8] Prakash, G., Saurav, N., & Kethu, V. R, "An Effective Undesired Content Filtration and Predictions Framework in Online Social Network", *International Journal of Advances in Signal and Image Sciences*, 2016; 2(2): 1-8.

[9] Olanrewaju, R. F., & Azman, A. W., "Intelligent Cooperative Adaptive Weight Ranking Policy via dynamic aging based on NB and J48 classifiers", *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 5, no.4, pp.357-365,2017

[10] Sulthana, R., & Ramasamy, S., "Context Based Classification of Reviews Using Association Rule Mining, Fuzzy Logics and Ontology", *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol.6, no.3,pp.250-255, 2017.

[11] Rao, R. R., & Makkithaya, K., "Learning from a Class Imbalanced Public Health Dataset: a Cost-based Comparison of Classifier Performance", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no.4, pp. 2215-2222, 2017.