Dynamic Integrated System for Detecting and Fixing Vulnerability Bugs

R. Anitha*1, M.V.Srinath2

1Department of Computer Science, Sengamala Thayaar Educational Trust Women’s College, Sundarakkottai, Mannargudi, Tamilnadu, India
2Sengamala Thayaar Educational Trust Women’s College, Sundarakkottai, Mannargudi, Tamilnadu, India

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.

Copyright © 2018 Institute of Advanced Engineering and Science. All rights reserved.

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 effective 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)**

$E \leftarrow$ entropy

**Step 1:** for $i = 0$ to $A$ do

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

end for

**Step 2:** for $j = 0$ to $A$ do

end for

**Step 3:** $\sum E_j = \sum_{k=1}^{Nod} R_k E_k$

end for

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

end for

**Step 5:** $E_j = \sum_{k=1}^{Nod} R_k E_k$

end for

**Step 6:** $Ig \leftarrow$ information gain

end for

**Step 7:** $Ba \leftarrow$ best attributes

end for

**Step 8:** for $j = 0$ to $A$.size-1 do

end for

**Step 9:** if $Ig_j > max$ then

max $= Ig_j$

end if

**Step 10:** $Ba = Ig_j$

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**

$Noc \leftarrow$ number of cluster

$Chs \leftarrow$ cluster heads

$T \leftarrow$ total

$Hp \leftarrow$ head probability

for $i = 0$ to $Noc$ do

$Chs_i = \text{select cluster head}$

end for

for $j = 0$ to $A$ do

$T + = Chs_{ij}$

end for

$Hp_i = T / A$.Size

end for

$Np \leftarrow$ node probability

$Ed \leftarrow$ euclidean distance

$M \leftarrow$ maximization

$Sh \leftarrow$ selected head

for $i = 0$ to $Noc$ do

for $j = 0$ to $A$ do

$T + = Chs_{ij}$

end for

end for

$Np_i = T / A$.Size

for $j = 0$ to $Noc$ do

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

end for
if $M > max$ then
    $max = M$
    $Sh = No_{c_j}$
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 $i = 0$ to $N_{od}$ do
    for $j = 0$ to $N_{od}$ do
        $Ed_{ij} = \sqrt{\sum_{k=1}^{A} (N_{od_{ijk}} - Ch_{s_{ijk}})^2}$
        if $Ed > min$ then
            $min = Ed$
            $T = N_{od_j}$
        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 dataset.

3. RESULT AND DISCUSSION

The software quality assurance checking on MPSO and EM algorithm is designed for reliable and effective bug predictions. It had been used to a number of clustering data. Figure 1 expresses the bug detection details with filtration reports.
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](image)

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 whose 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](image)

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 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