CHAPTER 6

ADAPTIVE FAULT TOLERANCE IN OPEN SOURCE SOFTWARE

6.1 INTRODUCTION

In this computerized world, software growth rate is increasing steadily. Some companies play an important role in this growth by developing and releasing new software to lead a leading position in the software development process. In order to make the product to be used by most of the people, the company releases their product as free software. The free software is an open source software which can be downloaded and utilized by all kinds of user at free of cost (i.e., no cost for downloading the software.)

The open source software is the software that sometimes may be developed without proper management. The quality and reliability of this software is not well designed. Since this software is an open source, the user is the only responsible person to solve all the problems faced in the software. The problem may arise while utilizing this software. In many instances the users of such softwares are not aware of its purpose, performance, usefulness and it is error-prone. Due to these problems, the users may get confused of whether to use this software further or leave it and search for an alternate one. They may also search for an alternate if they do not understand the errors occurred in the software. In such situation, the rate of people accessing this software becomes low.

In order to help the users to manage these problems, the research work proposes a fault tolerant technique. In this technique, the errors found in the open source softwares are automatically detected and reported. The fault tolerant is the technology that will tolerate the fault found in the software automatically and so the technology is termed as automated software.

In this work, fault tolerant technology is proposed for the open source software, to find the error rate automatically. Due to this automated technique, the rate of the people utilizing the software becomes more.

The aim is to propose a strategy to determine the performance of the open source software by deducing the error found in that software. This aim has to be achieved by deriving an algorithm to open and read the software and also to determine the errors automatically.

Thus this work implements an effective methodology in order to make the open source software useful by means of an automated fault tolerant technique to deduct and report the error found in the software, and to decrease the problem rate and manual effort.

6.2 METHODOLOGY PROPOSED FOR ADAPTIVE FAULT TOLERANCE IN OPEN SOURCE SOFTWARE

The aim of the proposed method is to develop a methodology with additional features of fault tolerance to make the process of executing the open source software, an automated process. This is to be made automated, in order to help the users utilizing the open source software. The fault tolerant method is implemented on open source software, It is executed and the efficiency of the software is determined by generating an error report.

The summary of this method is as follows: The client downloads open source software from the internet, to carry out the process. Upon downloading the software, the fault tolerance methodology is implemented on it its process is explained below.

6.2.1 Methodology

In this work, Java-based open source software is examined. In this kind of software, the downloaded software may be a JAR file, or a Class Program or it may be of some Packages. After downloading the software, the proposed method, processes in following steps:

First, it examines the software from the top-level of the downloaded software. That is, it starts counting the number of packages in the software. Upon completing the package counting, it then examines the number of classes under the package. Then it counts the number of methods in each of the classes. This counting is made by searching for the sub-folders in the software. This kind of analysis is used to determine and understand about the downloaded software.

The next step after the counting process is to compile the classes, methods, and packages to check for errors. The process is automated and so when the user wants to start the compilation process, they just have to initiate it. Then the process goes on automatically and compiles each and every classes, methods and packages continuously and check for errors. The automated software thus counts the number of errors occurred and then have to determine the error rate. Thus this automated process reduces the manual effort and the process time.

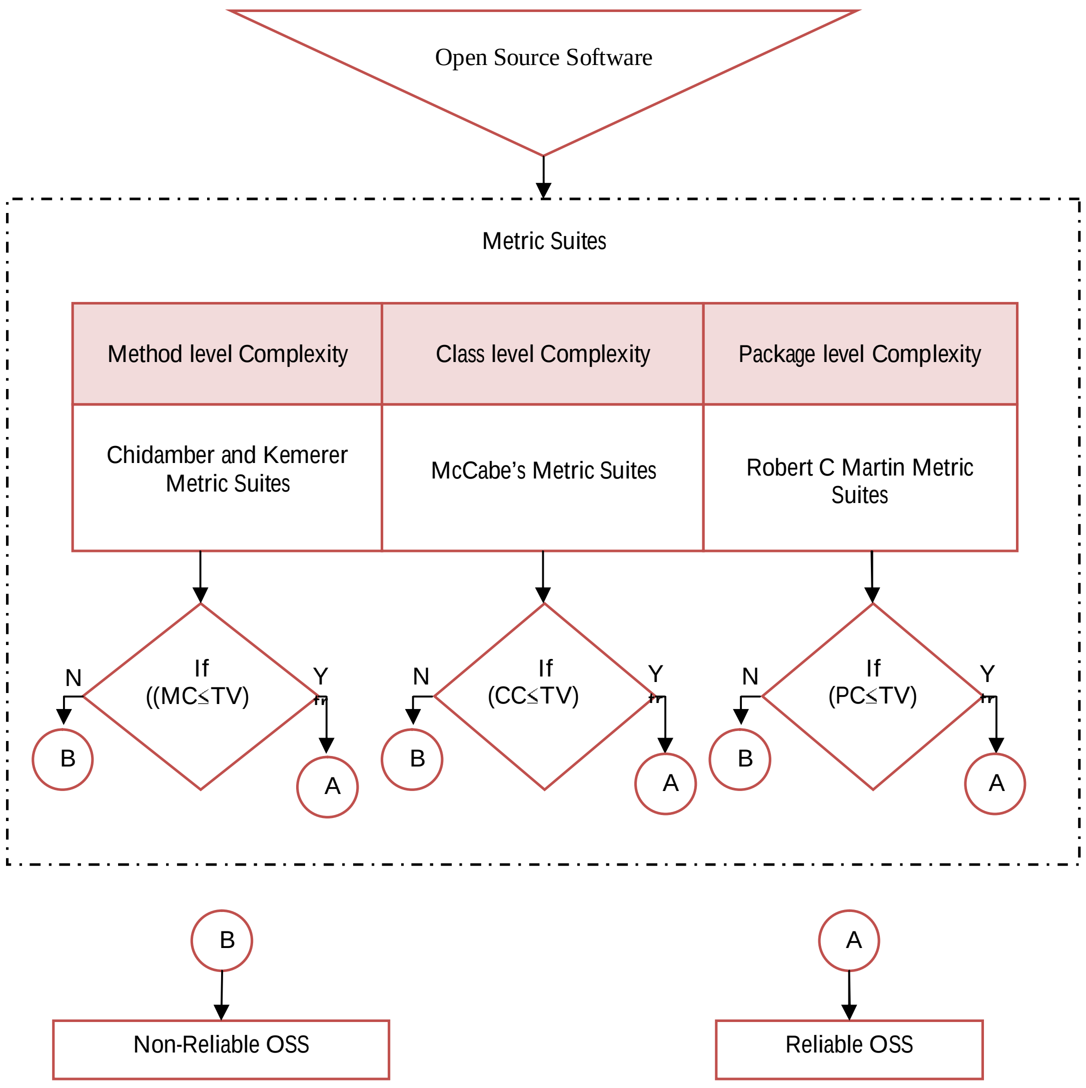

Figure 6.1 Methodology of Adaptive Faultolerance for OSS

The relevance of the metrics suites identified in the previous contribution of the research is implemented and the error rates are obtained for respective components as shown in Figure 6.1. The reliability based on adaptive fault tolerance is figured out with a comparison between threshold value and the metric value.

Based upon the error rate, the automated software helps the user to determine whether the errors are to be solved for further usage or alternative software has to be chosen to carry out their process. This is done by showing the number of errors, error report and the error rate. Also, based upon the error rate, the conclusion is made by the user.

Thus the automated software tolerates the fault occurred in the open source software, and helps the user to make the software useful.

6.2.2 Algorithm for Detecting Error Rate

The proposed Error Rate Detected (ECD) consists of an algorithm as shown in Figure 6.2. All the required validation processes will be taken into consideration by the proposed method. The following provides the algorithm used in this proposed method:

```
Start
    Open the Downloaded software.
    Count the Number of Packages, np
    Get np package names, p
    Count the Number of classes in each package, p as nc
    Retrieve nc Class names, c
    Count the number of methods in each classes, c as nm
    Get nm method names, m
    Read the method definition
    Compile the package.
    Determine the number of errors (syntax, type mismatch, etc.,), ne
    From ne and np, calculate the error rate.
        If error rate is low then
            Suggest that is executable
        Else
            Try for another package
        End if
    Error rate is shown.
End
```

Figure 6.2 Algorithm for detecting error rate

6.2.3 Explanation of the Algorithm

The algorithm works as follows: First the downloaded open source software is opened and the number of packages, classes and methods in it are counted. Then the algorithm retrieves the names of the packages, classes and methods and stores it. Then the compilation process is started and checks for the errors. The number of errors occurred are noted and the error rate is calculated and finally it is shown to the user. The Error rate is calculated with the errors identified over the Total count of methods, classes and packages.

Thus, this algorithm is used to determine the error rate and which in turn helps to rectify the errors or chose an alternate. The various modules of the algorithm are explained below.

TO DETERMINE THE ERROR RATE

```
// Count the Number of Packages
Search for the Package, P
If P > 0 then
        Pack_Count = Σ(P)
    Else
        Pack_Count = 0
End If
// Get the Package Name
For Pack_Index = 0 to Pack_Count
    Pack_Name [Pack_Index] = P [Pack_Index].Name
Next
//Count the Number of Classes in each Packages
For Pack_Index = 0 to Pack_Count
If Pack_Name [Pack_Index] = True then
    Search for the classes in that Pack_Name, C
  If C > 0 then
    Class_Count.Pack_Name = Σ(C)
    CCount++
  Else
    Class_Count.Pack_Name = 0
  End If
End If
```

```
**//Get the Class Name**
For Class_Index = 0 to Class_Count.Pack_Name
    Class_Name [Class _Index] = C [Class_Index].Name
    Next
Next
**//Count the Number of methods in each Classes**
For Pack_Index = 0 to Pack_Count
    If Pack_Name [Pack_Index] = True then
    For Class_Index = 0 to Class_Count.Pack_Name
    If Class_Name [Class_Index] = True then
            Search for the methods in that Class_Name, M
    If C > 0 then
            Method_Count.Class_Name = $\Sigma$(M)
            MCount++
    Else
            Method_Count.Class_Name = 0
    End If
End If
**//Get the Method Name**
For Method_Index = 0 to Method_Count.Class_Name
    Method_Name[Method_Index]=C [Class_Index].Name
    Next
Next
End If
Next
**//Compile and Determine the Error Rate**
Package_Count = $\Sigma$ (Pack_Count)
Class_Count = $\Sigma$ (CCount)
Method_Count = $\Sigma$ (MCount)
For pidx = 0 to Package_Count
    Compile all the classes and methods inside the Package(i).
If Error = true then
    Error++
End If
Next
TCount=Package_Count + Class_Count + Method_Count
**Error_Rate = (Error / TCount) * 100**
```

6.2.4 Results and Discussion

The experimental set up for this technique is carried out as follows: 2 to 5 Open Source Software are downloaded for experimental setup.

Without implementing the proposed method, all the process of opening the software, compiling the programs and deducing the errors are done manually. While doing manually, the process is too complicated and the performance becomes too slow. It takes much time to determine the error rate and to find the errors. The duration is calculated.

After that, the proposed technique is drawn on into the experimental setup. When this implementation is done, the automated software will start processing on the software automatically and will count the number of packages, classes and methods. Then it opens all the programs, compiles it, and determines the number of errors. Finally, it automatically displays the error rate to the user. Based on the error rate, the user makes decision about the further processing easily with less time consuming. The results are tabulated in the Table 6.1.

Table 6.1 Error report

Software Name	**Rhino 15R3**
Number of Packages	11
Number of Classes	160
Number of Methods	1577
Error Count	2
Error Rate	0.1144
Suggestion	Please verify the class path

Based on the error rate and the suggestion, the user makes the decision. Thus the implementation of the fault tolerant techniques is done successfully in this system.

6.3 CONCLUDING REMARKS

The objective to develop a methodology to tolerate the fault found in the Open Source Software automatically and to help the user to do their process easily is achieved in the research work. In this research work, an algorithm is proposed along with the methodology to explain about the process carried out while compiling the Open Source Software. The proposed work simplifies the user’s requirements by making use of the automated software to tolerate the fault. In this research, the technique is implemented on Java-Based Open Source Software as the scope is construed to it. For the other non-JAVA based open source software, some of the features like JAR files, class path, etc., are not applicable. Hence, Non-Java based Open source software is out of the scope of the research.