Research Article

A Large-Scale Study Of Fault Feature Extraction From Github Repository Using Data Science Techniques

P. Patchaiammal[1], G. Sundar[2], Dr. R. Thirumalaiselvi[3]

[1]Research Scholar, Bharath University, Chennaisarandsk1@gmail.com
[2]Research Scholar, Bharath University, Chennaisundarganapathy66@gmail.com
[3]Assistant Professor, Govt. Arts College (Men), Nandanam, Chennai

Article History Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: Nowadays data play a vital role in all the fields. Machine learning (ML) is a Data Science technique in which the past dataset can be used for future prediction. Fault is the result of the occurrence of an unexpected value in the place of expected value. Fault occurs in all fields, but the severity makes the difference in the industry. The severity of the fault is measured by its priority in the software industry. The unrectified fault may cause software failure. In this paper, thethree major development application-oriented software like website, mobile, and gaming are considered.Lots of programming languages areused for creation of these software. These applications are developed by all categories of projects like short, medium, and long in the software industry. This research works proved the necessity of fault taxonomy dataset by calculating rework range in this software creation. The fault feature data for this research work is collected from the GitHub public repository, which contains the language commit details in all categories of software industry.This work also helps in feature extraction by web scraping method to identify the rework in software development.

Key words: Machine Learning (ML), Data Science,Uniform Resource Locator (URL), Hyper Text Markup Language (HTML), GitHub Repository, Fault, Feature Extraction, Rework.

1. INTRODUCTION

Software is used in all set of works. An increase in the use of software means increase in fault. The fault reduction is used to achieve the qualified software. In order to gain the reduced fault, there is a need for fault taxonomy.Any software development model either traditional or continuous must increase the quality. The quality level will be reached only by testing and coding reviews. Due to lack of proper test strategies, the released software had poor quality. The programming language plays a major role in the software implementation. There are many languages used for various application developments such as scientific applications, websites, database, mobile applications, gaming etc. These are categorized by large, medium, and small sized projects. To increase the coding quality, the language related information should be associated with the application. For example, the Java Script language is used for website application, gaming, and also in mobile application. In such case, considering the previous fault factors helps to make the application a qualified one with localized fault. This helps to assure that the previous fault will not occur again. GitHub Repository has historical data information about many projects in different languages.

This research approach helps to find the necessity of fault taxonomy to predict the future fault in early development phase especially during implementation in order to save time and cost and to reduce rework in software production.

Data Science is responsible for the big data formation with necessary patterns. Machine learning is the one of the major techniques used to process and prepare the data in data science field. This quantitative study report is formed with the help of Machine Learning technique, a data science approach written in Python programming language. ML is a computer program, which is used to learn by example. ML algorithms are generalised from previously available historical data. The performance measures are displayed both in table as well as in graph. The table categorises the fault by three major size featuresas large, medium, and small and in further by based on the development of the application.

2. RESEARCH BACKGROUND

This research work helps to find out the necessity of early fault prediction and also insists the essentials of fault taxonomy in the mostly used implementation language of 3 major applications like website, gaming, and mobile.This work also shows the rework percentage and performance graph, which helps to identify the loss of time and cost effort in software project development.The previous work [9] is used to find the software fault prediction exploration by machine learning techniques. Features are major source of any research. Feature selection is the process, which is used to determine the relation between the input and the output of the problem whether it is relevant or not. This is done by forming the Hypothesis set in ML. The next work[3]is the stepping

stone to form the Hypothesis space for fault feature classification. This paper used four hypotheses testing against PROMISE dataset. The measured performance results showed that among four different types of hypotheses testing chi-square testing is more relevant for fault feature selection.Fitness function is the measure, which is used to identify that the result obtained is rightto the underlined problem or not. The next work [5] shows that the linear classifier is the best fitness function and is used to find only the reliable features from the feature samples taken from the population set. In this paper,genetic programming is used to find the root cause of the feature with the help of linear classifier fitness function. This paper work uses eclipsePROMISE dataset and from the 69 features only 29 is considered to predict the bug occurrence. Classified faults are predicted easier than unclassified one. The next work [7] helps to identify that the combined GKS algorithmic technique is the best one for fault classification. This background study report will help us to identify the necessity of fault taxonomy in order to check the fault database before testing and to reduce the risk post production fault to reduce the rework.

3. SOFTWARE FAULT FEATURE

Software implementation mistake found by the tester in the testing phase is the bug status of the software created. If that bug was accepted by the development team, then that error is known as defect. If the defect is present in the end user software, then it may cause failure. So,an unsolved defect is added in the documentation of the software manual. This time the same mistake is treated as feature. The features are formed to avoid software failure. The features are mostly associated to software development. An unsolved mistake in the software implementation phase affects the software qualityand also creates anomaly in the software. This anomaly is considered as fault feature, which is a distinguishable character in relation with the functional and performance of the software.

The above description says that the fault feature identification starts from bug. Therefore, in this research paper,closed and opened bugs in association with the major applications are collected from the GitHubrepository's commit history. The commit history is the collection of those feature faults, which are not solved and assigned against the implementation language. The three major application of development like website, gaming, and mobile are chosen. This research work is the first step to the formation of fault feature taxonomy to increase the software quality and also decrease the rework in post-production software.

4. METHODOLOGY DESIGN

The methodology of this research is divided by fourdata science phases as data detection, data preparation, data analysis, and data formulation. The method used to extract large amount of fault feature data is data science. In this research work, Pandas(Python data analysis) and matplotlib libraries associated with data science are used for performance analysis. Along with it feature extraction is done by data science method known as web scraping forthe collection of feature data. The urllib package is used for web scraping to get the details from the URL link. This package has request module and urlopen() function, which helps to pass the URL link to extract the details.

The necessary fault features related to this rework are language name, closed bugs, and opened bugsin association with the three different applications namely websites, mobile, and gaming. All these fault features are collected from the GitHubrepository commit history.

Phase 1: Fault Feature detection from GitHub Repository

In this phase the necessity of fault taxonomy is detected from the GitHub Repository site by searching in the URL link: https://GitHub.com/. Web scraping is done to detect and extract the commit details from the GitHub repository with the help of adding 'python library' files.

In GitHub Repository, there are both closed and opened bugs against each language. From those bugs top 10 languages used for the three major applications like website, gaming, and mobile are taken for web scraping. The various commit details in relation with opened bugs,closed bugs, and total bugs for 10 implementation languages are detected.

Phase 2: Data preparationusingpandas

The detected fault featuresfrom HTML code by web scrape as in the form of csv file using Pandas library. 'Pandas' is a high-performancepython software library, which is used here for the manipulation and the visualization of the tabular form of data.The visualized summary of data frame is used to easily identify the patterns related to the dataframe.

Phase 3: Data analysis by Formulation

The formation for the analytical model is done by calculating the sum of the closed bugs, opened bugs,total bugs along with CurrentReworkeffort(%), FutureReworkeffort(%) separately for all the three namely website application, mobile application, and gaming application.

1. Sum of Closed bugs

$$\text{Sum of Closed bugs}_{(\text{appln})} = \sum_{i=1}^{n}(\text{closed bugs}_i) \rightarrow (1)$$

where 'i' is language set from 1 to n with respect to the application('appln')known as website or mobile or gaming.

2. Sum of Opened bugs

$$\text{Sum of Opened bugs}_{(\text{appln})} = \sum_{i=1}^{n}(\text{Opened bugs}_i) + \sum_{i=1}^{n}(\text{closed bugs}_i) \rightarrow (2)$$

where 'i' is language set from 1 to n for the respective application('appln')known as website or mobile or gaming.

3. Sum of Total bugs

$$\text{Sum of Total bugs}_{(\text{appln})} = \sum_{i=1}^{n}(\text{Opened bugs}_i) \rightarrow (2)$$

where 'i' is language set from 1 to n for the respective application('appln') known as website or mobile or gaming.

4. CurrentRework Effort (%)

It is used to find the current rework effort needed for the respective application ('appln')known as website or mobile or gaming.

$$\text{CurrentReworkeffort}(\%)_{(\text{appln})} = \left(\frac{\sum(\text{Closed bugs})}{\sum(\text{Total bugs}))}\right) \times 100 \rightarrow (3)$$

5. FutureRework Effort%

It is used to find the current rework effort needed for the respective application ('appln') known as website or mobile or gaming.

$$\text{FutureReworkeffort}(\%)_{(\text{appln})} = \left(\frac{\sum(\text{Opened bugs})}{\sum(\text{Total bugs}))}\right) \times 100 \rightarrow (4)$$

Phase 4: Data analysis by performance measure

The analysis done by using three different csv files to calculate the total rework percentage for website, mobile, and gaming applications. The generalized rework reports are displayed in graph. These performance measures clearly shows the necessity of fault feature set to predict the fault earlier in implementation phase.

Phase 5: Data Extraction for future fault taxonomy

In this phase, the analysed data feature by rework performance, which is detected in phase 1 is extracted and saved by sqlite database file for future prediction taxonomy. This is done by using the web scraping operation website, mobile and gaming from GitHub Repository. In this phase, only the necessary commit details like language name, GitHub Repository URL link, number of opened bugs, and number of closed bugs related to each specific language are extracted with the help of python interface toolkit tkinter and the details are stored in csv file for the corresponding application.

5. IMPLEMENTATION

5.1 Feature Detection from GitHub repository

In this section, the data features related to website, mobile, and gaming are detected from GitHub Repository. The retrieved values from the corresponding GitHub Repository link are stored in csv form as database using tkinter toolkit.

5.1.1 Website feature deduction

Here the dropdown list box contains the entireURL link related to programing language of website application. The corresponding GitHub Repository link is shown in the textbox and the same details are saved into the 'website.csv' file as a fault feature with Opened and Closed bug count. In Diagram 1,the commit history related to one of the language java of website applications are shown with corresponding Closed and Opened bugs stored in GitHub repository.

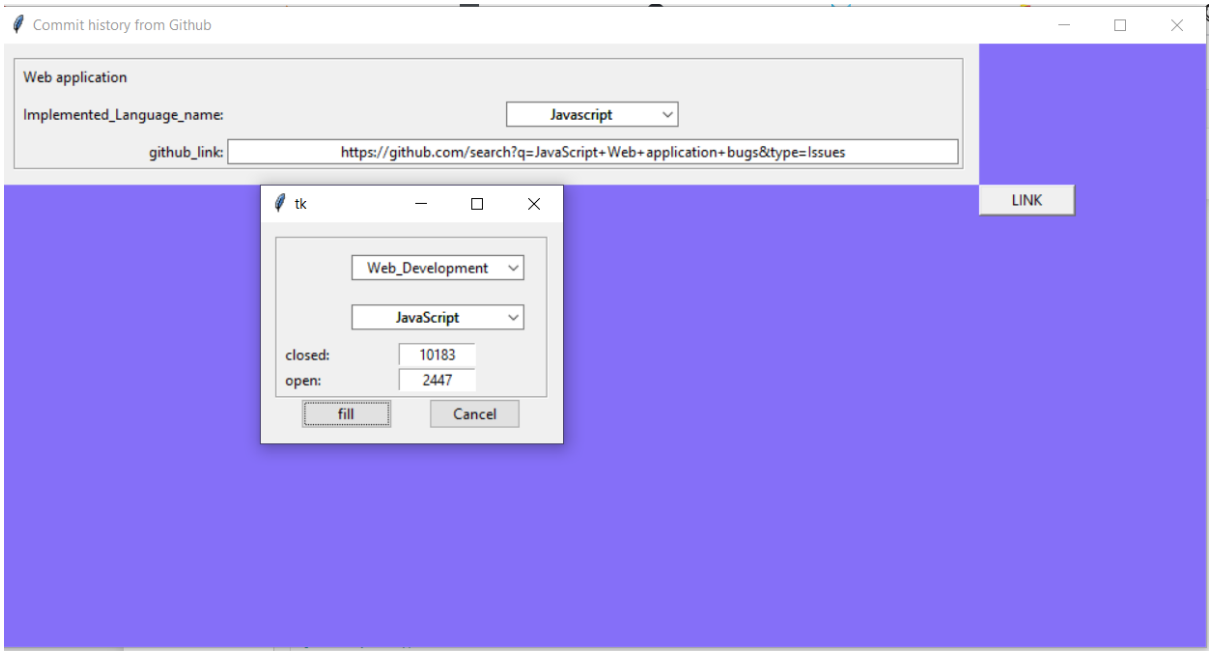

Diagram 1. Feature Detection of website application by web scraping data science analysis method

5.1.2 Mobile feature Detection

The dropdown list box contains the entireURL link related to programing language of mobile application. The corresponding GitHub Repository link is shown in the textbox and the same details are saved into the 'mobile.csv' file as a fault feature with Opened and Closed bug count. In Diagram2, the commit history related to one of the language Swift of mobile applications are shown with corresponding Closed and Opened bugs are stored in GitHub repository.

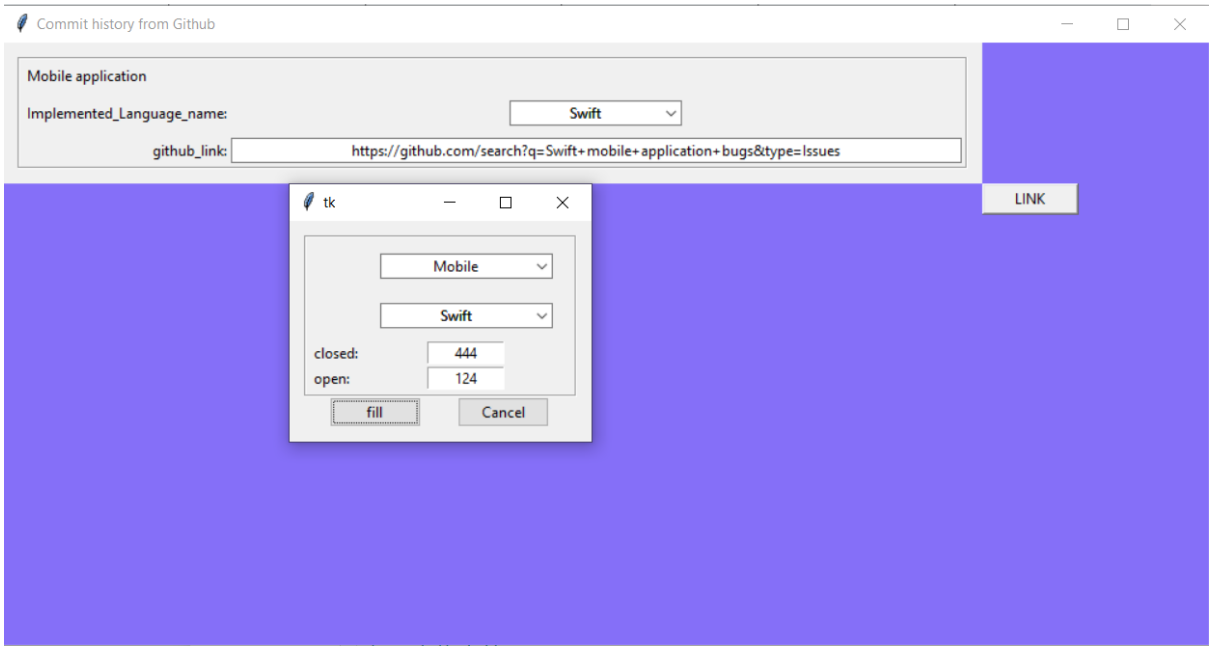

Diagram 2. Feature Detection of mobile application by web scraping data science analysis method

5.1.3 Gaming feature Detection

The dropdown list box contains the entire URL link related to programing language of mobile application. The corresponding GitHub Repository link is shown in the textbox and the same details are saved into the 'mobile.csv' file as a fault feature with Opened and Closed bug count. In Diagram 3, the commit history related to Swift language of mobile applications are shown with corresponding Closed and Opened bugs are stored in GitHub repository.

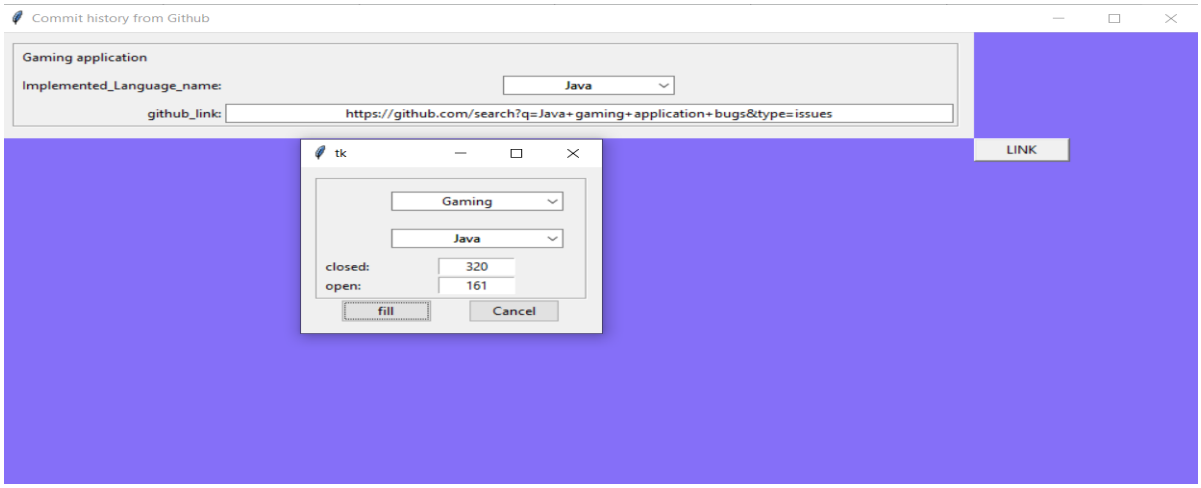

Diagram3. Feature Detection of gaming application by web scraping data science analysis method

5.2Data Preparation

In this section, the data is prepared and formed by 'Pandas' dataframe library. Data preparation is done by using the web scraping operation. The details of Closed and Opened commit history for three different applications with the help of df.dropna() command of pandas library, which is used to show all the filled data.

5.2.1 Website data preparation

The 10 best website development languages details are collected from GitHub by using the 'Pandas' python program. Tabular result form of the 'website.csv' is prepared for data extraction and the same is shown in the below Diagram4.

	languages	link	Closed	Opened
0	Python	https://github.com/search?q=Python+Web+applica...	11565	1993
1	Java	https://github.com/search?q=Java+Web+applicati...	7354	4483
2	JavaScript	https://github.com/search?q=JavaScript+Web+app...	10183	2447
3	Go	https://github.com/search?q=Go+Web+application...	17114	4995
4	Ruby	https://github.com/search?q=Ruby+Web+applicati...	1524	464
5	Dart	https://github.com/search?q=Dart+Web+applicati...	607	172
6	PHP	https://github.com/search?q=PHP+Web+applicatio...	4538	1129
7	Scala	https://github.com/search?q=Scala+Web+applicat...	311	114
8	HTML	https://github.com/search?q=HTML+Web+applicati...	17912	6792
9	Kotlin	https://github.com/search?q=Kotlin+Web+applica...	599	117

Diagram 4. Website data preparation using Pandas

The above listed details is the result of the python Pandas command df.dropna() of website.csv file. The result shows the languages list as Python, Java, JavaScript, Go, Ruby, Dart, PHP, Scala, HTML, and Kotlin along with their respective URLlink.The table also contains the Opened and Closed bugs of the languages. These are used to find the total rework effort in the concern language so as to form the feature fault taxonomy for early fault prediction in the website application development.

5.2.2 Mobile data preparation

The 10 best mobile development languages details are collected from GitHubby using the 'Pandas' python program. Tabular result form of the 'mobile.csv' is prepared for data extraction and the same is shown in the below Diagram5.

	Languages	link	Closed	Opened
0	JavaScript	https://github.com/search?q=JavaScript+mobile+...	4752	873
1	Kotlin	https://github.com/search?q=Kotlin+mobile+appl...	112	66
2	C++	https://github.com/search?q=C%2B%2B+mobile+app...	3869	2290
3	C#	https://github.com/search?q=C%23++mobile+appli...	3869	2290
4	Python	https://github.com/search?q=Python+mobile+appl...	4173	522
5	PHP	https://github.com/search?q=PHP+mobile+applica...	1253	706
6	Swift	https://github.com/search?q=Swift+mobile+appli...	444	124
7	Objective-C	https://github.com/search?q=Objective-C+mobile...	245	217
8	JAVA	https://github.com/search?q=Java+mobile+applic...	2502	1336
9	HTML 5	https://github.com/search?q=HTML5+mobile+appli...	3152	262

Diagram5. Mobile data preparation using Pandas

The above listed details is the result of the python Pandas command df.dropna() of mobile.csv file. The result shows the languages list as JavaScript, Kotlin, C++, C#, Python, PHP, Swift,Objective-C, JAVA, and HTML 5 along with their respective URLlink. The table also contains the Opened and Closed bugs of the languages. These are used to find the total rework effort in the concern language so as to form the feature fault taxonomy for early fault prediction in the mobile application development.

5.2.3 Gaming data preparation

The 10 best gaming development languages details are collected by using the 'Pandas' python program. Tabular result form of the 'gaming.csv' is prepared for data extraction and the same is shown in the below Diagram6.

	language	link	closed	opened
0	C++	https://github.com/search?q=C%2B%2B+gaming+app...	248	223
1	C#	https://github.com/search?q=C%23+gaming+applic...	574	339
2	Java	https://github.com/search?q=Java+gaming+applic...	161	159
3	Python	https://github.com/search?q=Python+gaming+appl...	164	100
4	Lua	https://github.com/search?q=Lua+gaming+applica...	50	26
5	Objective C	https://github.com/search?q=Objective+C+gaming...	39	37
6	JavaScript	https://github.com/search?q=Javascript+gaming+...	200	50
7	HTML5	https://github.com/search?q=HTML5+gaming+appli...	24	18
8	C	https://github.com/search?q=C+gaming+applicati...	248	223
9	SQL	https://github.com/search?q=SQL+gaming+applica...	16	15

Diagram 6. Gaming data analysis using Pandas

The above listed details is the result of the python Pandas command df.dropna() of gaming.csv file . The result shows the languages list as C++, C#, Java, Python, Lua, Objective C, JavaScript, HTML 5, C, and SQL along with their respective URLlink. The table also contains the Opened and Closed bugs of the languages. These are used to find the total rework effort in the concern language so as to form the feature fault taxonomy for early fault prediction in the gaming application development.

5.3 Rework effort calculation

In this section, the formulation of step 3 of methodology design is used for the rework effort calculation. It includes two different rework calculations like current rework and future rework. The rework calculation is used

to analyse the need for the maintenance of the fault as a taxonomy for the future prediction in order to reduce the rework in the post production software as well as the identification of root cause of the newly formed fault.

5.3.1 Website application Rework effortcalculations

Rework effort calculation of 10 languages with respect to website application is calculated by the python program using tkinter. The Current Rework Effort Percentage is calculated for Closed Bugs against Total Effort and the Future Rework Effort is calculated for Opened Bugs against Total Effort. The corresponding result is displayed by using 'Pandas' data frame.

```
df = pd.read_csv("website.csv")
```

```
df.dropna()
```

	languages	Closed	Opened	Total Effort	CurrentReworkeffort_Percentage	FutureReworkEffort_Percentage
0	Python	11565	1993	13558	85.300192	14.699808
1	Java	7354	4483	11837	62.127228	37.872772
2	JavaScript	10183	2447	12630	80.625495	19.374505
3	Go	17114	4995	22109	77.407391	22.592609
4	Ruby	1524	464	1988	76.659960	23.340040
5	Dart	607	172	779	77.920411	22.079589
6	PHP	4538	1129	5667	80.077642	19.922358
7	Scala	311	114	425	73.176471	26.823529
8	HTML	17912	6792	24704	72.506477	27.493523
9	Kotlin	599	117	716	83.659218	16.340782

Diagram 7. Website Rework effort calculation using pandas

5.3.2 Mobile application Rework effort calculations

Rework effort calculation of 10 languages with respect to mobile application is calculated by the python program using tkinter. The Current Rework Effort Percentage is calculated for Closed Bugs against Total Effort and the Future Rework Effort is calculated for Opened Bugs against Total Effort. The corresponding result is displayed by using 'Pandas' data frame.

```
df = pd.read_csv("mobile.csv")
```

```
df.dropna()
```

	language	closed	opened	Total Effort	CurrentReworkeffort_Percentage	FutureReworkEffort_Percentage
0	C++	248	223	471	52.653928	47.346072
1	C#	574	339	913	62.869660	37.130340
2	Java	161	159	320	50.312500	49.687500
3	Python	164	100	264	62.121212	37.878788
4	Lua	50	26	76	65.789474	34.210526
5	Objective C	39	37	76	51.315789	48.684211
6	JavaScript	200	50	250	80.000000	20.000000
7	HTML5	24	18	42	57.142857	42.857143
8	C	248	223	471	52.653928	47.346072
9	SQL	16	15	31	51.612903	48.387097

Diagram 8. Mobile Rework effort calculation using pandas

5.3.3 Gaming application Rework effort calculations

Rework effort calculation of 10 languages with respect to gaming application is calculated by the python program using tkinter. The Current Rework Effort Percentage is calculated for Closed Bugs against Total Effort and the Future Rework Effort is calculated for Opened Bugs against Total Effort. The corresponding result is displayed by using 'Pandas' data frame.

```
df = pd.read_csv("gaming.csv")
```

```
df.dropna()
```

	language	closed	opened	Total Effort	CurrentReworkeffort_Percentage	FutureReworkEffort_Percentage
0	C++	248	223	471	52.653928	47.346072
1	C#	574	339	913	62.869660	37.130340
2	Java	161	159	320	50.312500	49.687500
3	Python	164	100	264	62.121212	37.878788
4	Lua	50	26	76	65.789474	34.210526
5	Objective C	39	37	76	51.315789	48.684211
6	JavaScript	200	50	250	80.000000	20.000000
7	HTML5	24	18	42	57.142857	42.857143
8	C	248	223	471	52.653928	47.346072
9	SQL	16	15	31	51.612903	48.387097

Diagram 9. Gaming Rework effort calculation using pandas

5.4 Performance measure graph

The performance measure graph is drawn with the help of matplotlib data science tool. The graph shows the performance measures of the three different applications website, mobile, and gaming.

5.4.1 Website Performance measure

The performance measure of the website application is measured by using the three different measures of graph. All these measures are by drawn by using matplotlib.pyplot library of python.

The diagram 10(a) shows the Opened bugs related with website application. The graph shows the bug values of top 10 languages of website. The corresponding bugs of the languages are shown by dot in the graph.

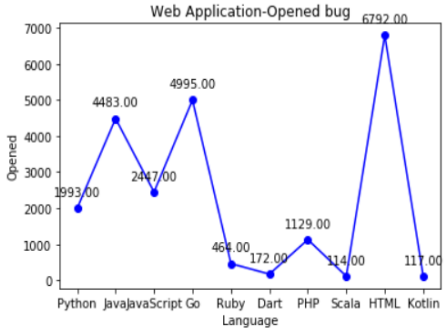

Diagram 10(a) Graphical representation of Opened bugs in website application of 10 languages

The diagram 10(b) shows the Closed bugs related with website application. The graph shows the bug values of top 10 languages of website. The corresponding bugs of the languages are shown by dot in the graph.

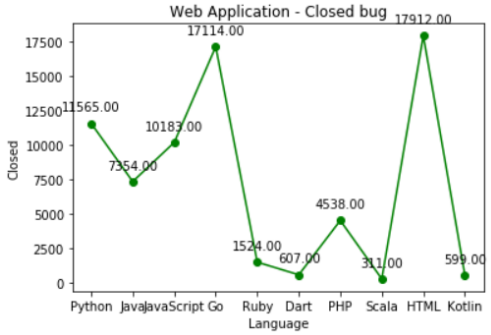

Diagram 10(b) Graphical representation of Closed bugs in website application of 10 languages

The diagram 10(c) shows the rework effort of both current and future related with website application. The graph shows the both current and future rework effort values of top 10 languages of website. The blue line shows the CurrentReworkeffort_Percentage. The green line shows the FutureReworkeffort_Percentage.

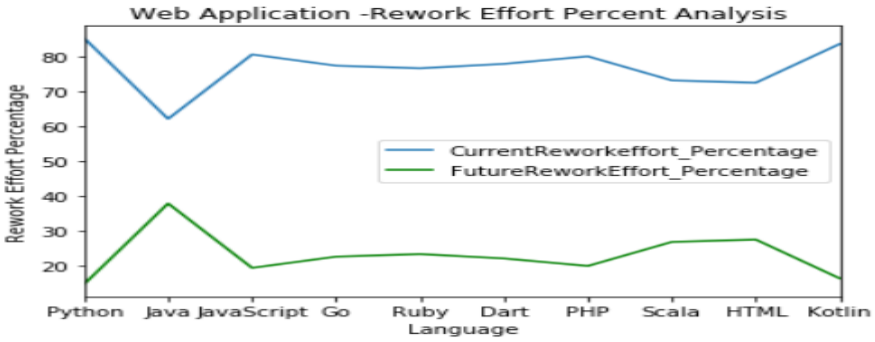

Diagram 10(c) Graphical representation of Rework effort percentage analysis measure in website application of 10 languages

The diagram 11(a) shows the Opened bugs related with mobile application. The graph shows the bug values of top 10 languages of mobile. The corresponding bugs of the languages are shown by dot in the graph.

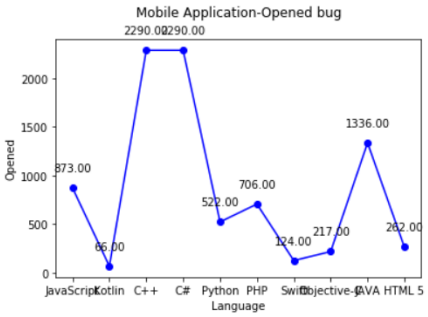

Diagram 11(a) Graphical representation of Opened bugs in mobile application of 10 languages

The diagram 11(b) shows the Closed bugs related with mobile application. The graph shows the bug values of top 10 languages of mobile. The corresponding bugs of the languages are shown by dot in the graph.

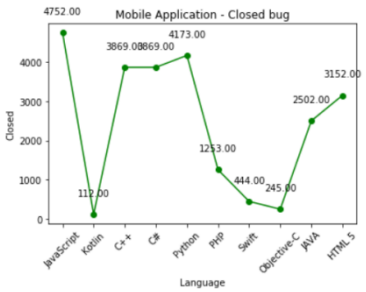

Diagram 11(b) graphical representation of Closed bugs in mobile application of 10 languages

The diagram 11(c) shows the rework effort of both current and future related with mobile application. The graph shows the both current and future rework effort values of top 10 languages of mobile. The blue line shows the CurrentReworkeffort_Percentage. The green line shows the FutureReworkeffort_Percentage.

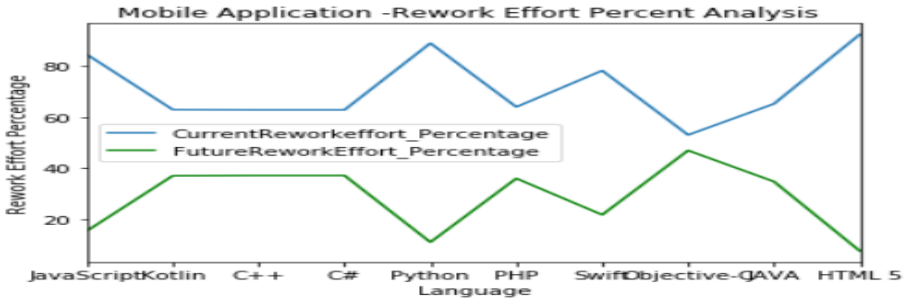

Diagram 11(c) graphical representation of Rework effort percentage analysis measure in mobile application of 10 languages

The diagram 12(a) shows the Closed bugs related with gaming application. The graph shows the bug values of top 10 languages of gaming. The corresponding bugs of the languages are shown by dot in the graph.

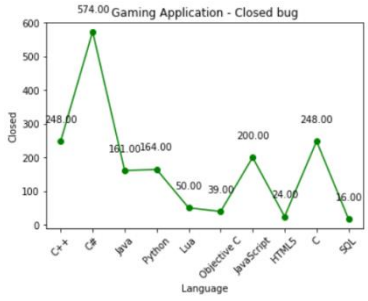

Diagram 12(a) graphical representation of Closed bugs in gaming application of 10 languages

The diagram 12(b) shows the Opened bugs related with mobile application. The graph shows the bug values of top 10 languages of mobile. The corresponding bugs of the languages are shown by dot in the graph.

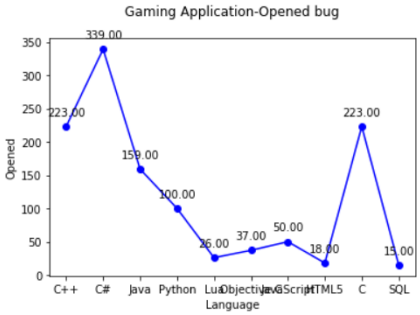

Diagram 12(b) Graphical representation of Opened bugs in gaming application of 10 languages

The diagram 12(c) shows the rework effort of both current and future related with gaming application. The graph shows the both current and future rework effort values of top 10 languages of gaming. The blue line shows the CurrentReworkeffort_Percentage. The green line shows the FutureReworkeffort_Percentage.

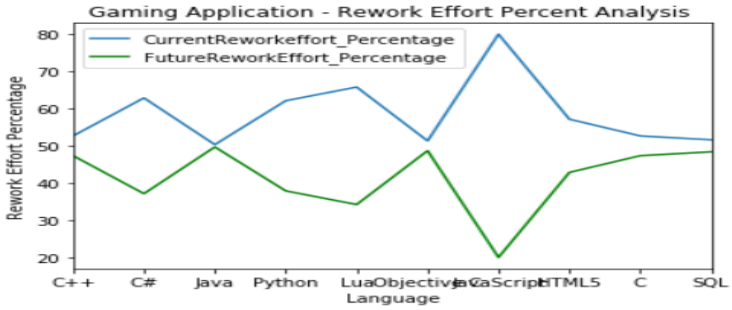

Diagram 12(c) Graphical representation of Rework effort percentage analysis measure in gaming application of 10 languages

The graph effort percentage proves that there is need for fault taxonomy in order to reduce the rework in software development. The corresponding bugs are collected and formed as fault taxonomy for the future prediction of fault in order to reduce rework.

5.5 Data Extraction for future fault taxonomy

The fault feature data is extracted as bugs from the corresponding URLlink from the GitHub Repository. These features are stored as sqlite database file for future usage. The stored database file of first 20 bugs is shown in the diagram 13. From the resultant file it is cleared that some of the bugs are repeated for more than one language. The bugs of 17,18 and 19 are repeated. From the repetitionit is proved that some of the bugs are repeated. So, by forming fault taxonomy with the bug details will helps to identify the repeated bugs of all the language which will also help to reduce the rework effort.

```
import pandas as pd
dfff1=pd.read_csv('C:\\Users\\Sar\\bf1.csv')
```

```
dfff1.head(20)
```

	Unnamed: 0	bug
0	0	C#9 language features
1	1	Memory unsafe languages, C/C++
2	2	refactor C omni completion in Vim9 script
3	3	Language server crashes on C++20 coroutines
4	4	Lifts upper bound on language-c
5	5	Ports based on rust ?
6	6	C bugs
7	7	0 bugs and Vulnerabilities
8	8	Add a source generator for a Maths language to...
9	9	Language specific bugs
10	10	Fixing C/C++/Objective-C classifications
11	11	Unit tests, core bugs
12	12	Known Bugs and Todo's
13	13	Use of conditional macros inside a macro invoc...
14	14	Implement -preview=complex
15	15	Enhancements to expression language
16	16	Celluloid poor video and wrong language, VLC Ok
17	17	Support for BUGS, JAGS, and Stan languages
18	18	Support for BUGS, JAGS, and Stan languages
19	19	Support for BUGS, JAGS, and Stan languages

Diagram 13.Fault taxonomy formed using GitHub URLlink

6. CONCLUSION

Even though popular planning and requirements are done, some of the software products are failed due to implementation fault. This paper provides the performance analysis of bug commit history with respect to the three different applications websites, mobile, and gaming taken from the GitHub repository. Even though the same languages are used each application the bugs are repeated. These bugs are to be corrected again by rework to avoid the failure. The current rework is calculated with respect to Closed bugs. The future rework is calculated with respect to Opened bugs. The calculated results are tabulated. Both of these graphical and tabular result shows that there is a need of classified bug taxonomy set to reduce the rework in software post-production. The fault taxonomy set is formed as the result of the feature extraction. This taxonomy will be used in future project

of same category to predict the bugs earlier in order to reduce rework and also to used recognise the root cause of the newly formedfault in the software development.

7. REFERENCES

1. GitHub Repository: https://api.GitHub.com/
2. Shih, Ya-Fang et al. (2017). "Deep Co-Occurrence Feature Learning for Visual Object Recognition." In Proc. Conf. Computer Vision and Pattern Recognition
3. P. Patchaiammal, R. Thirumalaiselvi "Genetic Evolutionary Learning Fitness Function (GELFF) for Feature Diagnosis to Software Fault Prediction" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue-11S, September 2019 Published By: Blue Eyes Intelligence Engineering & Sciences Publication Retrieval Number: K123309811S19/2019©BEIESP DOI: 10.35940/ijitee.K1233.09811S19
4. Ahmed Mateen, MarriamNazir and Salman Afsar Awan, "Optimization of Test Case Generation using Genetic Algorithm (GA)," International Journal of Computer Applications (0975 – 8887) Volume 151 – No.7, October 2016.
5. P. Patchaiammal, R. Thirumalaiselvi "Enrichment of Fault Features by Forming ML Hypothesis" Test Engineering and Management.January - February 2020 ISSN: 0193 - 4120 Page No. 6276 - 6288 Article Info Volume 82 Page Number: 6276 – 6288 Publication Issue: January-February 2020 Published by: The Mattingley Publishing Co., Inc.
6. Siegrist, Kyle. "Hypothesis Testing - Introduction". www.randomservices.org. Retrieved March 8, 2018.
7. P. Patchaiammal, R. Thirumalaiselvi "GKS Algorithmic Technique for Early Defect Prediction (GKS: A Genetic Feature K-means Clustering with Support Vector Machines)" International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 01, 2020 ISSN: 1475-7192 DOI: 10.37200/IJPR/V24I1/PR200282
8. Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs, René Just, DarioushJalali, Michael D. Ernst ISSTA '14, July 21–25, 2014, San Jose, CA, USA
9. P. Patchaiammal, R. Thirumalaiselvi "Software Fault Prediction Exploration Using Machine Learning Techniques" International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7 Issue-6S3 April, 2019 Published By: Blue Eyes Intelligence Engineering & Sciences Retrieval Number: F1022376S19/19©BEIESP Publication
10. 10.https://towardsdatascience.com/human-pose-estimation-with-stacked-hourglass-network-and-tensorflow-c4e9f84fd3ce

11. https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018

12. https://commons.wikimedia.org/wiki/File:Convolution_arithmetic_-_Same_padding_no_strides.gif

13. Torch, http://torch.ch/

14. Keras, https://keras.io/

15. Deeplearning4j, https://deeplearning4j.org

16. https://en.wikipedia.org/wiki/Precision_and_recall

17. https://bugs.eclipse.org/bugs/buglist.cgi?quicksearch=functional

18. https://www.bugzilla.org