International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Enhanced Object oriented metrics based Software Quality Analysing for Software Project

[a]Dr.T.Genish, [b]T.Nagaraj

[a]Assiociate Professor, Department of Computer Science, Karpagam Academy of Higher Education
[b]PG Student, Department of Computer Science, Karpagam Academy of Higher Education

A B S T R A C T

The main aim of this paper is to create an analysis tool to find the quality of the software. The user can identify whether their program is a quality one or not. The project contains 5 modules, which are as follows Software project selection module, LOC (Lines of Count) calculation module, Methods and attribute extraction, Quality analysis and report. The first module in the project is the program/code selection process, which the user needs to test. The user can give any program file or software files to test the quality. The next module is calculating LOC ie, line of code calculation which calculates total number of lines used in the program and finding unused lines from that. This module extracts total functions/methods used in the program, using this process; user can get the details of the objects, variables and methods. Based on the above modules, the quality of the software has been tested. This gives the overall quality of the program in percentage. The report gives the graphical representation of the quality verification data. And also list of tests made by the user will be stored in the database for future references.

Keywords: Lines of Code, Attribute extraction, Software fault.

1. Introduction

Improving unwavering quality of the ideal programming is quite possibly the most searched out exploration territories in computer programming. Programming engineers lay accentuation on planning dependable programming, so that inadequately planned programming can be recognized in the starter phases of the Software Development life cycle (SDLC) to try not to convey inferior quality programming item to the partner. In this manner, programming quality goes about as a significant factor in deciding the dependability of programming. In this way, there is a requirement for plan of forecast models to anticipate shortcoming inclined modules or classes in programming created dependent on item situated advancement system**.**

2. Methodology

MODULE DESCRIPTION

1.Software project selection module:

The first module in the project is the program/code selection process, which the user needs to test. The user can give any program file or software files to test the quality.

2.LOC (Lines of Count) calculation module:

The next module is calculating LOC ie, line of code calculation which calculates total number of lines used in the program and finding unused lines from that.

3.Methods and attribute extraction

This module extracts total functions/methods used in the program, using this process; user can get the details of the objects, variables and methods.

4.Clone detection using operation code sequences:

This module finds the duplication between the software's using the operation code sequences. Using this module, the admin can detect duplicate projects and programs easily.

5.Quality analysis

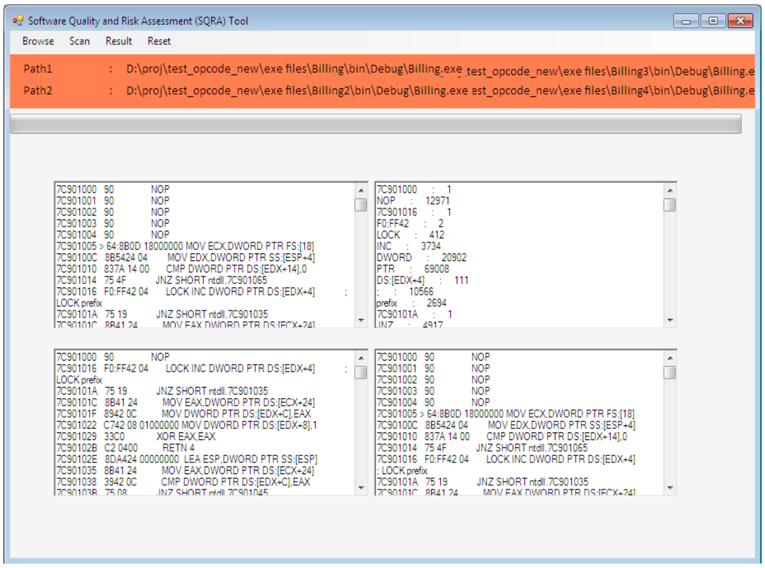
Based on the above modules, the quality of the software has been tested. This gives the overall quality of the program in percentage. The user can check whether the given program is efficient or not. Using this module two or more program will be compared with time and use of unique code features.

6.Report.

The report gives the graphical representation of the quality verification data. And also list of tests made by the user will be stored in the database for future references.

The purpose of this proposed system is detection of code duplications and visualizing duplicate information in a significant way. Detecting duplicate code fragments can significantly decrease the time and effort therefore the maintenance cost. Duplicate code detection can be achieved via analyzing the source code of given software system. Detection of code in softwares, which constitute code duplicate, is a non-trivial task because it is neither predictable nor obvious under what condition the duplicate code may appear. As a result, detailed analysis of software is a must to be able to detect duplicate code. It extracts the code form the executable file and checks the duplication. The main aim of this project is to create an analysis tool to find the quality of the software. The user can identify whether their program is duplicate or not.

Some software defects leads to failure only when certain local or non-local program interactions occur. Such interactions are modeled by the closely related concepts of information flows, program dependences, and structure and program segments. Those concepts are important in testing because they reflect interactions between different program elements. And there is a huge need to analyze the quality of the software. Existing software evaluation systems are only depends on the test cases where the tester should enter the expected and actual results. So the system needs to convert the code into machine language to get the structure of the program. Then the details will be verified by several testing and compilers.



```
Path1    :  D:\proj\test_opcode_new\exe files\Billing\bin\Debug\Billing.exe test_opcode_new\exe files\Billing3\bin\Debug\Billing.e
Path2    :  D:\proj\test_opcode_new\exe files\Billing2\bin\Debug\Billing.exe est_opcode_new\exe files\Billing4\bin\Debug\Billing.e
```

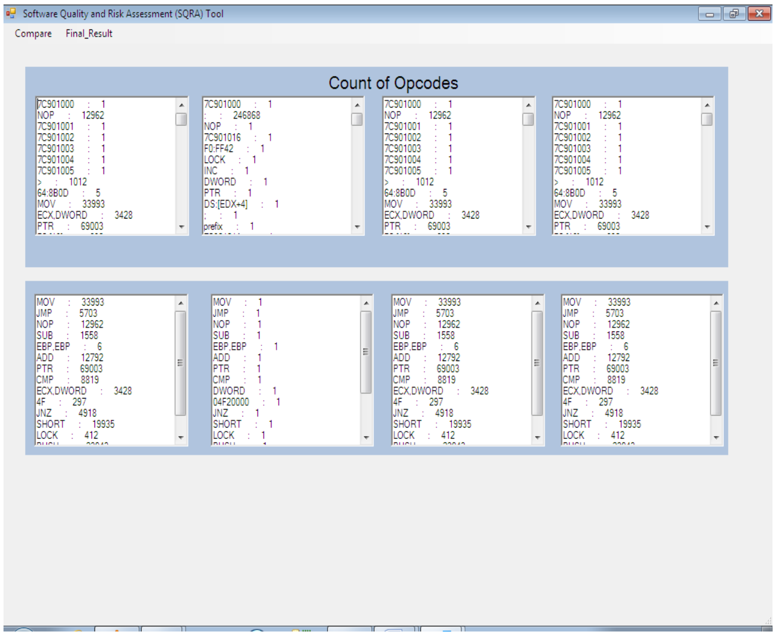
```
7C901000   90          NOP
7C901001   90          NOP
7C901002   90          NOP
7C901003   90          NOP
7C901004   90          NOP
7C901005 > 64:8B0D 18000000 MOV ECX,DWORD PTR FS:[18]
7C90100C   8B5424 04       MOV EDX,DWORD PTR SS:[ESP+4]
7C901010   837A 14 00      CMP DWORD PTR DS:[EDX+14],0
7C901014   75 4F         JNZ SHORT ntdll.7C901065
7C901016   F0:FF42 04      LOCK INC DWORD PTR DS:[EDX+4]        ;
LOCK prefix
7C90101A   75 19         JNZ SHORT ntdll.7C901035
```

```
7C901000    :   1
NOP    :   12971
7C901016    :   1
F0:FF42    :   2
LOCK    :   412
INC    :   3734
DWORD    :   20902
PTR    :   69008
DS:[EDX+4]    :   111
;    :   10566
prefix    :   2694
7C90101A    :   1
```

```
7C901000   90          NOP
7C901016   F0:FF42 04      LOCK INC DWORD PTR DS:[EDX+4]        ;
LOCK prefix
7C90101A   75 19         JNZ SHORT ntdll.7C901035
7C90101C   8B41 24         MOV EAX,DWORD PTR DS:[ECX+24]
7C90101F   8942 0C         MOV DWORD PTR DS:[EDX+C],EAX
7C901022   C742 08 01000000 MOV DWORD PTR DS:[EDX+8],1
7C901029   33C0          XOR EAX,EAX
7C90102B   C2 0400         RETN 4
7C90102E   8DA424 00000000  LEA ESP,DWORD PTR SS:[ESP]
7C901035   8B41 24         MOV EAX,DWORD PTR DS:[ECX+24]
7C901038   3942 0C         CMP DWORD PTR DS:[EDX+C],EAX
```

```
7C901000   90          NOP
7C901001   90          NOP
7C901002   90          NOP
7C901003   90          NOP
7C901004   90          NOP
7C901005 > 64:8B0D 18000000 MOV ECX,DWORD PTR FS:[18]
7C90100C   8B5424 04       MOV EDX,DWORD PTR SS:[ESP+4]
7C901010   837A 14 00      CMP DWORD PTR DS:[EDX+14],0
7C901014   75 4F         JNZ SHORT ntdll.7C901065
7C901016   F0:FF42 04      LOCK INC DWORD PTR DS:[EDX+4]
; LOCK prefix
7C90101A   75 19         JNZ SHORT ntdll.7C901035
```

Fig-1 OP-code verification



```
7C901000   :  1
NOP   :  12962
7C901001   :  1
7C901002   :  1
7C901003   :  1
7C901004   :  1
7C901005   :  1
>   :  1012
64:8B0D   :  5
MOV   :  33993
ECX,DWORD   :  3428
PTR   :  69003
```

```
7C901000   :  1
:   :  246868
NOP   :  1
7C901016   :  1
F0:FF42   :  1
LOCK   :  1
INC   :  1
DWORD   :  1
PTR   :  1
DS:[EDX+4]   :  1
;   :  1
prefix   :  1
```

```
7C901000   :  1
NOP   :  12962
7C901001   :  1
7C901002   :  1
7C901003   :  1
7C901004   :  1
7C901005   :  1
>   :  1012
64:8B0D   :  5
MOV   :  33993
ECX,DWORD   :  3428
PTR   :  69003
```

```
7C901000   :  1
NOP   :  12962
7C901001   :  1
7C901002   :  1
7C901003   :  1
7C901004   :  1
7C901005   :  1
>   :  1012
64:8B0D   :  5
MOV   :  33993
ECX,DWORD   :  3428
PTR   :  69003
```

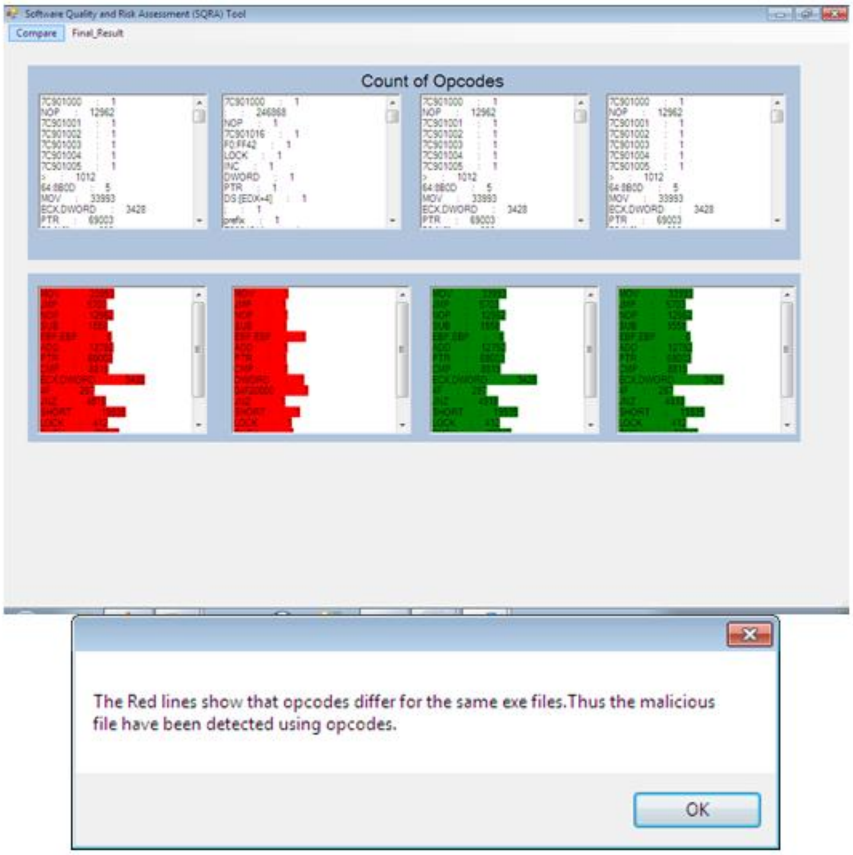
```
MOV   :  33993
JMP   :  5703
NOP   :  12962
SUB   :  1558
EBP,EBP   :  6
ADD   :  12792
PTR   :  69003
CMP   :  8819
ECX,DWORD   :  3428
4F   :  297
JNZ   :  4918
SHORT   :  19935
LOCK   :  412
```

```
MOV   :  1
JMP   :  1
NOP   :  1
SUB   :  1
EBP,EBP   :  1
ADD   :  1
PTR   :  1
CMP   :  1
DWORD   :  1
04F20000   :  1
JNZ   :  1
SHORT   :  1
LOCK   :  1
```

```
MOV   :  33993
JMP   :  5703
NOP   :  12962
SUB   :  1558
EBP,EBP   :  6
ADD   :  12792
PTR   :  69003
CMP   :  8819
ECX,DWORD   :  3428
4F   :  297
JNZ   :  4918
SHORT   :  19935
LOCK   :  412
```

```
MOV   :  33993
JMP   :  5703
NOP   :  12962
SUB   :  1558
EBP,EBP   :  6
ADD   :  12792
PTR   :  69003
CMP   :  8819
ECX,DWORD   :  3428
4F   :  297
JNZ   :  4918
SHORT   :  19935
LOCK   :  412
```

Fig-2 count of OP-code



```
7C901000   :  1
NOP   :  12962
7C901001   :  1
7C901002   :  1
7C901003   :  1
7C901004   :  1
7C901005   :  1
>   :  1012
64:8B0D   :  5
MOV   :  33993
ECX,DWORD   :  3428
PTR   :  69003
```

```
7C901000   :  1
:   :  246868
NOP   :  1
7C901016   :  1
F0:FF42   :  1
LOCK   :  1
INC   :  1
DWORD   :  1
PTR   :  1
DS:[EDX+4]   :  1
;   :  1
prefix   :  1
```

```
7C901000   :  1
NOP   :  12962
7C901001   :  1
7C901002   :  1
7C901003   :  1
7C901004   :  1
7C901005   :  1
>   :  1012
64:8B0D   :  5
MOV   :  33993
ECX,DWORD   :  3428
PTR   :  69003
```

```
7C901000   :  1
NOP   :  12962
7C901001   :  1
7C901002   :  1
7C901003   :  1
7C901004   :  1
7C901005   :  1
>   :  1012
64:8B0D   :  5
MOV   :  33993
ECX,DWORD   :  3428
PTR   :  69003
```

The Red lines show that opcodes differ for the same exe files.Thus the malicious file have been detected using opcodes.

fig-3 Results

3. Conclusion

The proposed SQA system used OpCode with sequence and frequency analysis patterns generated by disassembling the inspected executable files to extract features from the inspected files, the system extracts the methods and body content to get the opcodes. OpCode extraction algorithms with sequence analysis are used as features during the testing process with the aim of identifying unknown malicious code and quality of the program. The proposed system performed an extensive evaluation using a test collection comprising more than 30 methods with 4 assembly files. The evaluation consisted of three experiments. In the first experiment, this found that the frequency and sequence representation then it will count the opcodes and provides the results. The proposed tool has been named as SQA , which provides a time consuming test results for malicious data finding. Every application has its own merits and demerits. The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Changing the existing modules or adding new modules can append improvements. Further enhancements we can integrate and check any type of application quality through our software more efficient manner

References

1. Ahad, A., Yalavarthi, S. B., & Ali Hussain, M. (2016). A new approach for integrating social data into groups of interest doi:10.1007/978-81-322-2755-7_24 Retrieved from www.scopus.com
2. Chandra Prakash, V., Sastry, J. K. R., Kantharao, V., Sriharshini, V., Sriram, G., & Ganesh, C. H. V. S. (2017). An expert system to assess memory power of a student for selection of a suitable career. Journal of Advanced Research in Dynamical and Control Systems, 9(Special Issue 6), 309-321. Retrieved from www.scopus.com
3. Chaitanya Krishna, B., Sai Mounish, P., Vinay Kumar, U., &Divya Mallika, A. (2018). Online business site quality assessment. International Journal of Engineering and Technology(UAE), 7, 838-840. Retrieved from www.scopus.com
4. Sharma, N., &Yalla, P. (2016). Software engineering and natural language processing- how can they be together? International Journal of Software Engineering and its Applications, 10(12), 389-396. doi:10.14257/ijseia.2016.10.12.32
5. Preetham, C. S., Siva Ganga Prasad, M., Pratap Reddy, N., &SashankaTeja, L. (2016). Outage probability analysis of amplifyandforward and decodeandforward dual hop relaying with hardware defects. Journal of Theoretical and Applied Information Technology, 86(2), 290-298. Retrieved from www.scopus.com
6. Naga Malleswari, D., Suresh Babu, S., Moparthi, N. R., Mandhala, V. N., & Bhattacharyya, D. (2017). Hash based indexing in running high dimensional software systems. Journal of Advanced Research in Dynamical and Control Systems, 9(Special Issue 6), 34-43. Retrieved from www.scopus.com
7. Chandraprakash, V., Sastry, J. K. R., Sravani, G., Manasa, U. S. L., Khyathi, A., & Harini, A. (2017). Testing software through genetic algorithms – a survey. Journal of Advanced Research in Dynamical and Control Systems, 9(Special Issue 2), 1607-1633. Retrieved from www.scopus.com
8. M Praneesh and Jaya R Kumar. Article: Novel Approach for Color based Comic Image Segmentation for Extraction of Text using Modify Fuzzy Possiblistic C-Means Clustering Algorithm. IJCA Special Issue on Information Processing and Remote Computing IPRC(1):16-18, August 2012. Published by Foundation of Computer Science, New York, USA.