TEACHING AUTOMATED TEST DATA GENERATION TOOLS FOR C, C++ , AND JAVA PROGRAMS

Hitesh Tahbildar[1] and Plabita Borbora[2] G. P. Khataniar[3]

Department of Computer Engineering and Application, Assam Engineering Institute, Guwahati, Assam 781003, India
tahbil@rediffmail.com,(plabita.borbora,drkhataniar)@gmail.com

ABSTRACT

Software Testing cost 50% of the software development budget. Using automated tools we can reduce the testing cost. There are huge numbers of automated testing tools on the web. Teaching automated testing tools to students is an important topic for computer engineering students. Even faculties are not ware about the software testing tools.

In this paper, a platform is provided for easy way of learning some testing tools which is very helpful for teaching as well as practical software testing. Most popular languages like C, C++, Java languages testing tools are taken for illustration. Finally, different testing tools are compared with some criteria to understand the facilities and limitations available on testing tools. Our work can help students and faculty members to understand different testing tools and how to use the appropriate one for a programming language. We are using windows platform for our experiments.

Keywords:

NCSS, CCN, unit testing tools

1. INTRODUCTION

The Prime objective of academic institutions is to produce students who can get a job easily with their knowledge. Statistics shows that most of our output gets job in software industry. Software is the main component that is learned by every computer Engineering student in our organization. They do their project to learn how to develop a software. But it is seen that although students learn software development in academic institutions they hardly follow real software development life cycle where 50% or more effort should be spent on software testing [8] to release a software in the market[10]. The students emphasize only on software design and coding part. Testing part of software development life cycle is taken casually. Whatever software they develop they do not do systematic testing either manually or using automatic testing tools. During the development phase of a software, developers have to face with several challenges and consider various scenarios[5]. To tackle the challenges without compromising on both performance and robustness of the software the developer have to look the following issues

- Platform independence
- Stability of code.

DOI : 10.5121/ijcsit.2013.5115

- Code optimization for performance(speed and space)
- Coding standard and guidelines.
- Array bound checking.
- Dead code, unused variables
- Software consistency for 32 bit and 64 bit system
- Exception handling etc.

If such issues are not detected in the early stages of the development life cycle, the customers will be unsatisfied.

Ultimately it will effects to the reputation and profile of the organization. Since manual testing is very costly, Use of automatic testing tools are essential to reduce the cost of software so that we can compete with global market. An automatic testing tool of testing offers automation wizards and commands of its own in addition to providing task recording and replay capabilities. It is self controlling and self moving processes[3]. An Automatic testing means less people are needed to test system, and often testing can be carried out with one person managing the emulation of hundreds of terminals on a network. Further, we observed that students are very much confuse while using the software testing tools. There are huge number of software testing tools can be found on the web. Converging few important tools from the large domain of automated tools is also a big work and it takes lot of time. Even most of the faculties are confuse to provide proper guidelines to study selected tools for practice to students. There are very few organization who can spent money for testing tools. It is seen that most of the educational institutes have no automated testing tools. We want to provide guidelines to study automated testing tools with minimum effort. Our work outcome will help students to get exposure of automated testing tools and they can learn easily automated testing tools. Even we try to test our software using some automatic tool but it is not clearly known which tools are best for which programming language and selecting a tool for testing is difficult. In literature we can not find clear cut idea to select a tool for testing a particular software written in C/C++ and Java.

In this paper, we want to explore different open source testing tools for unit testing and try to compare some tools using some criteria[1]. We use both quality criteria and functional iteria[1]. Commonly found quality criteria for testing are functionality, reliability, usability, efficiency, maintainability, portability, vendor support, licensing and pricing etc [1].

1.1 Why Open Source Tools

After evolution of software testing market, a number of testing tools are now available ranging from simple unit testing to end to end life cycle management. It is owing to the expansion of market the cost of testing tools have been rising continuously. This is motivating academic Institution to seek open source testing tools which are less expensive from a total cost of ownership perspective and often equivalent functionality. Open source testing tool is available in source code form. Students can understand the source code. Certain other rights and source code are provided under open source license. Normally these rights are reserved for copyright holders. Open source license permits users to study, change, improve and redistribution of the software.

1.2 Open source Tool for Static analysis

This type of tool analyzes the software without executing program built from that software. Mostly the analysis is performed on some version of the source code and in the other cases some form of the object code. In static analysis, Code is examined over all possible behaviors that might arise during run time. Code of each unit is validated against requirements of the unit by reviewing the code. The static unit testing code is reviewed by applying techniques-Inspection

and Walkthrough. In Inspection each step is checked against pre-determined criteria. We have a list of commonly found errors. In Walkthrough the author leads the team through a manual or simulated execution of the product using pre-defined scenarios Normally, static analysis tools detect logical errors of following types which are not detected by compiler such as memory leaks and common logic errors. It detects bugs such as a warning about an unused argument, issues related to constructors, exception handling checking, Array Bound checking, Method checking etc. Automated tools can scan the entire code that helps to find the weaknesses earlier in development life cycle for reducing the cost to fix. It permits weaknesses to be found earlier in the development life cycle, reducing the cost to fix.

1.2.1 Limitations of Static Analysis

- It is time consuming if conducted manually.
- Automated tools do not support all programming languages.
- Automated tools produce false positives and false negatives.
- There are not enough trained personnel to thoroughly conduct static code analysis.
- Automated tools can provide a false sense of security that everything is being addressed
- It does not find vulnerabilities introduced in the runtime environment.

1.3 Open source Tool for Dynamic analysis

In dynamic analysis, a program unit is actually executed and its outcomes are observed. One observes some representative program behavior, and reach conclusion about the quality of the system. It require test data/test input and gives the logical answer either pass or fail with feedback. It is able to detect dependency that is not possible in static analysis. For example, dynamic dependencies using reflection, dependency injection, polymorphism. Temporal information can be collected by dynamic analysis.

1.3.1 Limitations of Dynamic Analysis

- It is much more complex to work with.
- It cannot guaranteed the full coverage of the source codes as it runs based on user interaction or automatic tests.

We decided to select three open source unit testing tools (Static and dynamic)for each of the programming languages C/C++ and Java, since C/C++and Java are mostly used in software industries and our students does their final year project in these programming languages. Our paper facilitates students to use and experiment testing tools with very less effort.

The rest of the paper is organized as follows**:** The section 2 presents a survey of related automated testing tools with reference to programming languages C, C++, Java. The section 3 describes some Selected Automated Testing Tools. The section 4 shows a comparison of Selected Automated Testing Tools. Finally in section 5 we conclude how our work can help students and faculties to learn and teach testing tools.

2. RELATED WORK

In literature a number of software testing methods and testing tools are found. There is no testing methods/tools that can generate test data for any programs. In general case it is an undecidable problem[11]. Mustafa, Al-Qutaish, Muhairat provide a classification of different testing tools on the basis of different types of testing methods and software products[6]. Although the paper consider different types of application software but it fails to explain the concept of static and

dynamic analysis. Moreover it provide no emphasize on open source tools which are becoming much popular nowadays. Thomas Hofer in [13] comparison of some static tools both open and commercial using parameters like Installation, Configuration, Support, Reports, Errors found, Project support. In this paper, Dynamic tools are not taken care. Chin etal survey focuses on some testing tools of Fortran languages[8]. Kaur and Kumari, in [9] made a comparative study of testing tools. The paper shows clearly the comparison with graph but they covers only two tools and language is also not specified.. T.Illes etal did software testing tools evaluation considering different criterion[1]. Dave Wittry[14] explains JUNIT tool for motivation, usefulness for the students and teachers. The paper is good for getting practical knowledge. Thomas Ball explains the dynamic analysis concepts[2]. The List of tools for different Languages with their respective websites are in Appendix A.

3.SOME SELECTED AUTOMATED TESTING TOOLS

From large domain of testing tools found in the web we selected some testing tools based on some criteria and language as shown in figure 1.

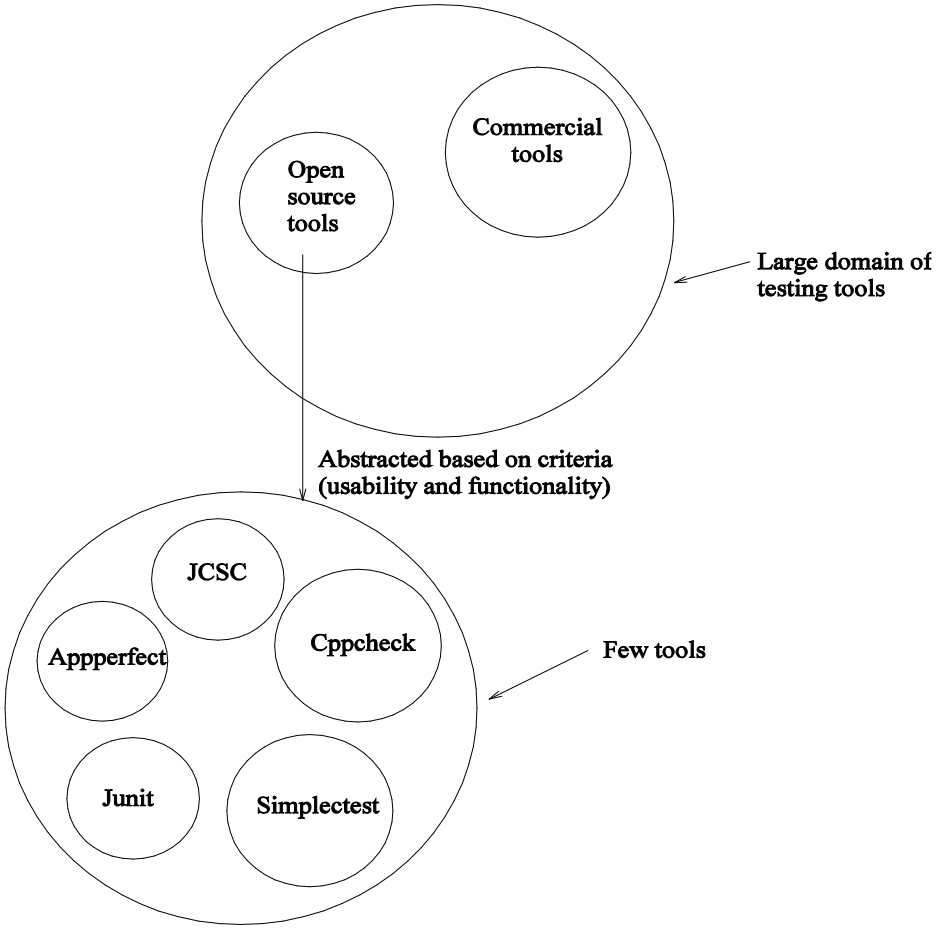

Figure 1: Selection of Testing tools

The selected tools as shown in figure 1, are both static and dynamic tools for programming languages C , C++, and Java.

Static Analysis Tool

- Cppcheck (for C and C++)
- AppPerfect(for Java)
- JCSC (For Java)

Dynamic Analysis Tool

- JUNIT using Netbeans (for java
- JUNIT using Dr. Java (for java)
- SimpleCtest (For C and C++)

4. FOR C AND C++ PROGRAM

4.1 Cppcheck: Static analysis tool for C/C++ (code analyzer)

Developer: Daniel Marjamaki
Cppcheck only detects the types of bugs that the compilers normally fail to detect. The goal is no false positives. Cppcheck can work on any platform that has sufficient CPU and memory. We can check both files and folder using cppcheck[19]. It can run on cross platform.

4.1.1 How To Use

First we have to download the tool from http://sf.net/projects/cppcheck/ .After that the following steps have to be followed.

- Select the C/C++ file.
- Save the file under Cppcheck directory.
- Open Cppcheck and select the option check.
- Select particular file or directory.

In another way we can run cppcheck from DOS prompt.

1. click start button -> run -> type cmd
2. Move to the Cppcheck directory
3. Type Cppcheck gui space filename with c or cpp extension

If errors are found it will display on the output section. Reports are provided in XML format. In case of folder checking, first open the Cppcheck. After that select check option -> directory . Select particular directory. All error files under this directory will be opened . By clicking each file, we can view the errors or warning. Normally the following bugs Cppcheck can detects [18] 64-bit portability, Auto Variables, Boost usage, Bounds checking, Check the code for each class, Exception Safety, Check input/output operations, Leaks (auto variables), Match assignments and conditions, Memory leaks (address not taken), Memory leaks (class variables), Memory leaks (function variables), Memory leaks (struct members), non reentrant functions, ,Null pointers, Obsolete functions, Standard Template Library (STL) usage, Uninitialized variables, Unused functions, Unused Var, Using postfix operators.

Note: Cppcheck is rarely wrong about reported errors. But there are many bugs that it cannot detect.

4.1.2 Illustration with example

Consider the following program

```
#include <iostream.h>
#include <conio.h>
void main()
int i,a[10];
clrscr();
for(i=0;i<12;i++)
cin>>a[i];
for(i=0;i<12;i++)
cout<<a[i]<<"\n";
```

In the above program it displays the errors in following way-

Severity	Summary
Error	7 Buffer access out of bounds :a
Error	9 Buffer access out of bounds :a

4.2 Simplectest (dynamic unit testing tool for C and C++)

It is a simple but effective testing framework for C and C++ projects, without having too much overhead. It can be run in Cygwin (MS Windows), OS Portable (Source code to work with many OS platforms).In other testing framework C++ is strictly types and does not have support for reflection. Moreover we have to write out the test functions more than once or we use a scripting language to generate the list of tests. Simplectest is based on macros. "Simplectest" is designed to be used at the same time as implementation of our application. It provides a simple framework for running code, checking assertions and producing a report at the end[20]. We donot need to use a scripting language like Perl to generate the lists of tests to run. Also we don't need to try and dance with complex features like templates.

4.2.1 How To Use

For installing Simplectest no make/configure is required. All we need to do are included the macro definitions in our test header files and include the header file in our test program like:

```
#include " tests.h"
```

Steps for running:

1. The test program is first saved under C:/tcc/Bin
2. After that run cmd and move to the tcc directory
3. After compiling the program test the program by typing the program name.

4.2.2 Illustration with example

```
#include "tests.h"
Start the overall test suite

ASSERT NOT EQUALS FLOAT(12.5==12);
```

```
A new group of tests, with an identifier
START_TEST("simple")
We then write the tests we want to check
ASSERT(1 == 1);
ASSERT EQUALS FLOAT(1.5, 1.5);
END_TEST()

START_TEST("fail")

These tests will fail, and we will get output.

ASSERT(10 == 10);
ASSERT EQUALS FLOAT(1.2, 1.2);

Lets give a description of the test, before it
fails - this will be printed out instead.
TEST("we expect this test to fail. (3==2)");
ASSERT(3 == 3);
END_TEST()

// End the overall test suite

Output of the program

Test runs: 2
Passes: 5
Failures: 0
```

4.3 For Java Program

4.3.1 Java Coding Standard checker(JCSC): (Static unit testing tool for Java)

Developer: 1999-2005 by Ralph Jocham

JCSC is a powerful tool to check source code against a highly definable coding standard and potential bad code. The standard covers naming conventions for class, interfaces, fields, parameter etc . The existence of correct JavaDoc can be enforced. Apart from that, it finds weaknesses in the the code , potential bugs like empty catch/finally block, switch without default, throwing of type 'Exception'etc. JCSC supports over 100 enforceable rules. It can run in Windows (with and without cygwin), Linux, Mac OS X After testing programs , we observe that the following types of general checks are done. correct class/interface header, line length, NCSS (non commenting source statements = real code) count for class, NCSS checking for method length, CCN (cyclomatic complexity number) checking for methods, tabulators in source code allowed , space after statement (if, else, while, ...) ie any formatting error, throwing of 'Exception' or only of subclasses types allowed etc. JCSC also perform Naming Checks like package, class, interface, abstract class, label, instance field, class (static) field, constant (final static) field, local variables, instance method name etc. JCSC displays the Metrics like " NCSS (non commenting source statements- real code) are being calculated for the whole project, individual classes and methods " CCN (cyclomatic complexity number - possible number of paths through a method) are generated for all methods and constructors

4.3.2 How To Use

First download the tool from the site http://jcsc.sourceforge.net. After that do the following steps

1. install J2SE in our system
2. extract JCSC into a folder and name it JCSC HOME
3. right click on Mycomputer icon, click properties->advanced->Environment variable-> new button(top one)-> JCSC HOME(type in top box)->in the next box paste the path copying from from the folder eg-../../JCSC HOME/bin---
4. In the bottom part (system variables),we have to edit the path.
5. Select the java program and compile to create class file.
6. Select start button -> run and type cmd
7. type cd and give one space and type the path "../../JCSC HOME/bin"
8. now type JCSC space filename extension name with java

4.3.3 Illustration with example

Consider the following program

```
import java.util.*;
import java.io.*;
public class TestWrite
public static void main(String[] args)throws FileNotFoundException
PrintWriter datafile=new PrintWriter("test1.txt");
String aLine = "Assam Engineering Institute";
datafile.println(aLine);
datafile.close();
```

For the above program the analysis by the JCSC will be

Violations : 7 nos.

Metrics ->

5:38:TestWrite.main(): NCSS - 5 CCN - 1

NCSS count : 8
Lines count : 16
Methods count : 1
Unit test class count :0
Unit test count : 0

4.4 AppPerfect Java Code Test : (static Java code analysis software)

Developer : AppPerfect Corporation

It analysis our Java and Java Server Pages (JSP) source code and applies about 750 Java coding rules. It is designed to perform two key tasks: Automate Java code review and Enforce Good Java coding Practices. We notice that during the development phase, testing and deployment, the cost of fixing problems grows exponentially. By conducting source code analysis and successfully identifying and correcting all such issues, developers can eliminate the risk and potential costs

early in the software development cycle. Locating and fixing performance problems during source code development time is the cheapest way to resolve problems [22].

4.4.1 How To Use

We can download demo tool from the following site

http://www.softpedia.com/get/Programming/Other-Programming-Files/Appperfect-Java-Code-Test.shtml. After downloading the tool, we can run the tool from Start -> Programs ->AppPerfect Java Code Test x.x.x -> AppPerfect Java Code Test. First we have to create a new project in AppPerfect Java Code Test by clicking File -> New... AppPerfect uses a project wizard for AppPerfect Java Code Test. Once a project is defined, we need to specify certain values related to the specific App Test for which we are creating the project. Listed below are the various properties which can be set using new project dialog.

- General
- Import
- Source (add the file to be test)
- Java Settings
- Environment (Specify the CLASSPATH and LIBRARYPATH)
- Target (to specify Server / Web Application or Local / Desktop Application)

Following categories are there in AppPerfect Java Code Test under which the rules have been grouped. We can also edit rules through rule manager - 64 bit machine Rules, Code Convention Rules, EJB Rules, Garbage Collection Rules, Hibernate rules, Internationalization Rules, J2ME Rules, Javadocs Rules, Java Metrics Rules, JSP Metrics Rules, JSP Rules, JUnit Rules, Naming Convention Rules, OOPs Rules, Optimization Rules, Portability Rules, Possible Errors Rules, Security Rules, Servlet Rules, Struts Rules, Unused Code Rules Under each rule different rules are available.

For more details one can view the following cite
http://www.appperfect.com/support/docs/java-code-test/index.html

Along with pre defined rule sometimes we can define our own custom rules. Once a project is defined, to analyze the project , select Project -> Run... from the main menu.. As the analysis progresses, the views display the result of analysis. Violations that are displayed can be autofix or Suppress violations. AppPerfect shows the metrics like Classes Metrics ,Class wise Blank Lines , Class wise Commented Lines, Method Complexity, Method Parameters ,Method , Nested Block Depth, Non-static attributes, Static attributes,Total Lines. AppPerfect can perform the followings Apply 750 Java coding rules, Auto fix over 180 types of violations, detailed Metrics Information, Informative and user friendly reports, Scheduling and Notification, Integration, ANT Integration and Command line execution.

AppPerfect Java code test has some Limitations like

- It Can only analyze 10 files
- 15 days trial
- Nag Screen

4.4.2 Illustration with example

Consider the following program

```
import java.util.*;
import java.io.*;
public class TestWrite
public static void main(String[] args)throws FileNotFoundException
PrintWriter datafile=new PrintWriter("test1.txt");
String aLine = "Assam Engg Institute";
datafile.println(aLine);
datafile.close();
```

For the above program the analysis by the appPerfect will be

violation(File wise) - 2 no. ->

1. declare_ imports_ in_ alphabetical_ order
2. Use_ buffered_ IO

In first case it shows the severity Low and category is code convent
In the second case it shows the severity critical and category is optimization

Violation(developer wise) -2 no.

Metrics' class - 1
Classwise Blank lines - 5
Methods - 1
Classwise line - 10
Nested Block depth - 1
Total lines - 12
Method complexity - 1
Method parameter - 1

Moreover AppPerfect also displays the analysis time.

4.5 JUNIT: (Dynamic Unit testing tool for Java)

Developers : Kent Beck, Erich Gamma, David Saff

Unit testing is commonly automated. Under the automated approach a developer could write another section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is done. They could also isolate the function to test it more rigorously. Using an automation framework, the developer codes criteria into the test to verify the correctness of the unit. JUnit has been important in the development of test driven development, and is one of a family of unit testing frameworks collectively known as XUnit that originated with SUnit. It is a regression testing framework. JUnit is linked as a JAR at compile-time. JUnit 4.x is a test framework which uses annotations to identify methods that are test methods. JUnit assumes that all test methods can be executed in an arbitrary order. Therefore tests should not depend on other tests [21]. JUnit tests allow us to write code faster while increasing quality. It is elegantly simple and takes less time. JUnit tests can be run automatically and they check their own results and provide immediate feedback. The tests can be organized into test suites containing test cases and even other test suites. To write a test with JUnit.

- Annotate a method with @org.junit.Test
- Use a method provided by JUnit to check the expected result of the code execution versus the actual result.

Using Junit we test different programs like array equality, summation, area of rectangle etc. using test data. Junit test and evaluate a program by executing data in real time. It tries to find errors in a program while it is running. Some of the key features of JUnit are annotation to identify the test methods(After JUnit 4), assertions for testing expected results, Test fixtures for sharing common test data, Test runners for running tests, aggregating tests using suites.

4.5.1 How to use

Using net beans IDE**:** The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible Java virtual machine (JVM). A pre-existing JVM or a JDK is not required. In netbeans test cases are created automatically when we select "create Junit Test" for particular program. Only test data should be given. JUnit unit testing framework can be integrated into DrJava.

4.5.2 Illustration with example

Consider the following program

```
public class Calc

int add(int a , int b)
return(a+b);

int multi(int a , int b)

return(a*b);
```

For dynamic testing of the above class, we can use netbeans. In netbeans Junit test cases for all methods are created automatically. We have to give the test data using junit test methods like assertTrue((String, Boolean),

Assert Equals(String, int, int) etc.inside the methods.

For example we can write assertNotSame(expResult, result) in the testadd()

```
@Test
public void testAdd()
Calc instance = new Calc();
int expResult = 30;
int result = instance.add(10,20);
assertNotSame(expResult, result)
```

The above test will generate output as pass.

4.6 DrJava:

Dr Java is a lightweight IDE for Java . It includes an intelligent program editor, an interactions pane for evaluating program text, a source level debugger, and a unit testing tool. Here also, Similar to netbeans, test cases are created automatically when we give input test data.

5 .DISCUSSION AND COMPARATIVE STUDY

5.1 Discussion

We experimented both static and Dynamic analysis tools. These tools are selected on the basis of usability and functionality criteria. Usability means understandability, learnability, operability and Functionality means suitability, accurateness, interoperability, compliance and security[17]. All these tools (for C/C++ and Java) support windows operating system and all are open source. Among the open source tools for static analysis of C/C++ under windows operating system that are available in the internet. Cppcheck is easy to install and easy to use. Cppcheck detects bugs which are not detected by the compiler like common logic error, memory leaks, issues related to constructor, array bound checking etc. Other static analysis open source tools for C/C++ neither support windows operating system nor easy to install. Some commercial tools are available for windows operating system. But here we discussed only open source tools. Moreover Cppcheck can be used in both GUI environment and command line environment. For Java programming two static analysis open source testing tools AppPerfect and JCSC(Java Coding Standard Checker) are used. These two tools can be downloaded easily from the mentioned website. AppPerfect support GUI environment and generic command line execution(non UI) from .bat or .sh file to execute tests but JCSC support command line interface only. Though AppPerfect has some limitation like full features demo version is available for 15 days only, test limited files(10 files),display nag sreen, advantages of this tool is that we can see the violations separately (Filewise and developer wise) with severity(high, medium and low) and categories. We can autofix (fix the violation) or suppress (ignore) violations . Violation which are suppressed may also be unsuppressed later so that they start appearing back in the violation view. Moreover AppPerfect displays the analysis time. Both AppPerfect and JCSC open source testing tools analysis the software according to their predefined rules. We can edit rules. According to experiment it is seen that JCSC gives more violation than the Appperfect in case of formatting errors. It has a graphical UI for easy rule configuration .JCSC gives the non commenting source statement(NCSS) count for class and cylomatric complexity number (CCN) that are not given by AppPerfect. Between these two tools AppPerfect has more facilities than the JCSC in the following since-

1. It support GUI and Command line.
2. More coding rule
3. Autofix and suppression
4. Generate report separately etc.

Finally we see that static analysis tools (for C/C++ and Java) that gives us output/information which are not detected by compiler. They analysis the source code according to their own rules. We donot require any input. Dynamic analysis require test inputs/ test data. Basically dynamic analysis tool generate testing framework where we will give input manually. This type of tools gives logical answer either as pass or fail with feedback. The difference of Simplectest testing tool from other C/C++ dynamic testing tool is that in other testing framework, C/C++ is strictly typed. Moreover we have to write the test functions more than once or we use a scripting language to generate the list of tests. But simplectest avoid too much overhead and provides a simple framework for running code, checking assertions and producing a report at the end. All testing can be completely independent of our actual code.

5.2 Comparative studies of selected testing tools

We compare the experimented/used testing tools Cppcheck, AppPerfect, JCSC, Junit and Simplectest. Different criteria are used for comparison. The table 1 shows our comparison.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

Now a days testing plays a important role in software development. It is seen that the majority amount of money of total software development cost are invested in software testing. In this paper, some static and dynamic analysis of open source

Criteria	Cppcheck	JCSC	AppPerfect	Junit	Simplectest
Type	Static	Static	Static	Dynamic	Dynamic
Language	C/C++	Java	Java	Java	C/C++
OS	Windows/Linux	OS independent	OS independent	OS independent	OS independent
Vendor Support	No	No	Yes	No	No
Licensing	GNU	General	Public license	GPL ASLA	General public license
Configuration	Automatic	Complex	Standard	Automatic	Simple
Coding rule User Interface Support	About 400 GUI and Command Line	Over 100 Command Line	About 750 GUI and Command Line	Not specific GUI and Command Line	Not specific GUI and Command Line
Project Support	Not Specific	Single file	10 files	Single file	Single file

Table 1: Comparison of Some Testing Tools

testing tools that are based on usability and functional criteria are explained so that our students can test their software easily that is done in final semester project. Since our most of the students develop software in C,C++ and Java, therefore we consider the static and dynamic analysis tools for C, C++ and Java. Finally we compare the selected tools using some parameter that help our students to understand different tools and use the appropriate one. By studying the tools, they can install and use the tool for different languages under windows environment. All the tools that are discussed broadly are very good open source tools for windows operating system. All are easy to use. It is also seen that if we want to test a software completely, both static and dynamic analysis tools are necessary.

6.2 Future Scope

Since we discussed the open source testing tools for windows operating system, our future research focuses on providing a survey on testing open source tools for other operating environment. Moreover we have studied only five tools(both static and dynamic) for windows operating system. Therefore one can study other different open source testing tools for windows environment and compare the tools using some common criteria so that students and faculties can use more appropriate testing tool.

REFERENCES

[1] T. Illes, A. Herrmann, B. Paech, J. Rtickert , Criteria for Software Testing Tool Evaliutation- A Task Oriented View, Technical Report, Institute for Computer Science, University of Heidelberg, Germany.

[2] Thomas Ball , The Concept of Dynamic Analysis, Technical Report, Bell Laboratories, Lucent Technologies.

[3] Rajiv Chopra , Software Testing A practical Approach, S.K. Kataria and Sons. Second Edition,2009

[4] Rahul Shende, Testing in 30+ open source tools, Mumbai, Shroff Publishers and distributors Pvt. Ltd. First Edition : September 2010.

[5] Jon Wiley Sons " Software Testing fundamentals, Methods, and Matrics"

[6] K. Mustafa, Rafa E, Al-Qutaish, M,I. Muhairat “Classification of Software testing ools Based on the Software Testing Methods' Second International Conference on Computer abd Electrical Engineering, Amman, Jordan, 2009.

[7] J. Edvardsson,, “A Survey on Automatic Test Data Generation," In Proceedings of the Second Conference on Computer Science and Systems Engineering(CCSSE'99), Linkoping, pp. 21-28 10/1999.

[8] L. S.Chin, D.J. Worth, and C. Greenough : "A Survey of software Testing Tools for Computational Science, “Software Engineering Group Rutherford Appleton Laboratory, LHarwellScience and Innovation Campus, Dideot.Oxford June 29,2007.

[9] Manjit Kaur and Raj Kumari : "Comparative Study Automated Testing Tools: TestComplete and QuickTest Pro", International Journal of Computer Application , Volume 24 No.1 June 2011.

[10] Hitesh Tahbildar and Bichitra Kalita : "Automated software test data Generation : Direction of Research " International Journal of Computer Science and Engineering Survey , Volume 2 No. 1, PP 99 - 120, March 2011.

[11] Hitesh Tahbildar and Bichitra Kalita : "Automated test data generation based on individual constraints and boundary value" IJCSI International Journal of Computer Science Issues , Vvol. 7, pp. 350-359, September 2010.

[12] Poston,R.M.; Sexton, M.P. : "Evaluating and Selecting testing tools" IEEE Software, olume: 9 Issue:3, May 1992, S.33-42.

[13] Thomas ofer : "Evaluating static source code Analysis Tools" IEcole Polytechnic , Lausanne,2007.

[14] Dave Wittry : "Automated Unit Testing with JUnit A tool for unit testing" design, lab grading,

[15] R. Tokar and S. Mankefors: " Survey on Testing and Reuse ", Sw STE'03, IEEE Conference, 0-7695-2047-2,2003

[16] Secologic : "Open Source Static Analysis Tools for Security Testing of Java Web Applications " An Analysis Document Version 1.0 - Dec. 2006

[17] Source Code Analysis Tools - Overview (commercial and free tools; with an focus on C/C++), Department of Homeland Security, 1/2006 https://buildsecurityin.us-cert.gov/daisy/bsi/ articles/tools/code /263.html

[18] en.wikipedia.org/wiki/Cppcheck

[19] cppcheck.sourceforge.net/

[20] simplectest.sourceforge.net/

[21] en.wikipedia.org/wiki/Junit

[22] John Wiley Sons- Software testing Fundamentals, Methods and Metrics

[23] Vaaraniemi, Sami: "The benefits of automated unit testing.",The Code Project. 9 November 2003. 29 April 2004 http://www.codeproject.com/gen/design/onunittesting.asp.

AUTHORS

Dr. H. Tahbildar Received his B. E. degree in Computer Science and Engineering from Jorhat Engineering College, Dibrugarh University in 1993 and M. Tech degree in Computer and Information Techno logy from Indian Institute of Technology, Kharagpur in 2000. He received his Ph.D in Computer Science and Application in 2012 from Gauhati University and his current research interests are Automated Software Test data generation of procedural and Object oriented programming, Program Analysis, E-Governance, He is working as HOD, Computer Engineering Department, Assam Engineering Institute, Guwahati, INDIA.

Mrs. P. Borbora: She received her B. E. degree in Computer Science and Engineering from Jorhat Engineering College, Dibrugarh University in 1993 and M.S. degree in Software Systems from BITS Pilani in the year 2013 and her current research interest is Software Testing, An alysis of software Tools. She is working as Lecturer (Sr. Scale), Computer Engineering Department, Assam Engineering Institute, Guwahati, INDIA.

Dr. G. P. Khataniar Received his B. E. degree in Computer Science and Engineering from Jorhat Engineering College, Dibrugarh University in 1992 and M. Tech degree in Computer Technology from Indian Institute of Technology, Delhi in 1999. He received his Ph.D in Computer Scien ce and Engineering in 2011 from Indian Institute of Technology, Guwahati and his current research interests are Distributed Systems, Object oriented programming and Testing. He is working as Lecturer(SG), Computer Engineering Department, Assam Engineering Institute, Guwahati, INDIA.