I. J. Education and Management Engineering, **2020, 5, 28-34**
Published Online October 2020 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijeme.2020.05.04

Experimental Comparison of Mutation Testing Tools for C Sharp Language

Tariful Alam
Student, Department of Computer Science, American International University-Bangladesh (AIUB), 408/1, Kuratoli, Khilkhet, Dhaka 1229, Bangladesh
E-mail: alam.tariful@gmail.com

A. G. M. Zaman
Assistant Professor, Department of Computer Science, American International University-Bangladesh (AIUB), 408/1, Kuratoli, Khilkhet, Dhaka 1229, Bangladesh
E-mail: agmzaman@gmail.com

Received:14 June 2020; Accepted: 15 July 2020; Published: 08 October 2020

Abstract: Mutation testing is a popular software testing technique, that inject artificial faults into the program and requires test cases to reveal these faults. In this paper, an experimental comparison is analyzed for different types of mutation testing tools in C Sharp language in .NET framework. Different mutation testing tools are giving different mutation score for a program. The objective of this paper is to investigate why the mutation score is different for different tools, and the scope of generating mutants depending on different types of operators. Four tools, such as, Visual Mutator, Cream, Ninja Turtles, and Nester are selected and applied to a program and analyze the outcome. Among these four mutation testing tools, Visual Mutator is better because of its higher mutation score, and it generates mutants for both common and uncommon standard operators and object level operators.

Index Terms: Software Testing, Mutant, Mutation Testing, Mutation Score, Mutation Testing Tools, Test Case, C Sharp Language.

1. Introduction

Software testing is the process to check the quality of the software by using a set of test cases [1]. Mutation testing is the most powerful testing technique that is used to check the effectiveness of the test cases by checking the mutant adequacy score [2, 3]. The faulty version of the program is called the mutant, and test cases are used to detect those mutants via mutation testing tools. There are many different mutations testing tools for different programming languages.

This paper is focused on comparison between different testing tools for C Sharp language in .Net Framework. Four mutation testing tools are analyzed in this experiment, such as, Visual Mutator, Cream, Ninja Turtles and Nester. These four tools are mostly used for .net framework. The most important difference among them is the operator coverage for generating mutants. There are two types of operators, standard, and object-oriented operators. Depend on the operator's coverage mutants are generated by different tools for a program, and test cases are used to kill these mutants. Finally, best tool is selected based on the mutation score, which is calculated for standard common and uncommon object level operators.

2. Background Study

This section highlights the basic concepts of mutation testing technique.

2.1 Mutation

Mutation testing is a fault-based testing technique and consider as white box testing for unit testing which provides a testing criterion called the mutation adequacy score. This mutation adequacy score can be used to measure the quality, reliability and performance effectiveness of a test set in terms of its ability to detect faults [1,4,5]. In the year of 1971, Richard Linkon et al. first wrote a paper on fault tolerant of computer program [6]. In [4,7,8] their research objective demonstrates the whole mutation testing process, such as, how mutation testing execute, how mutant program create, what to change in a mutant program, mutation testing techniques and mutation score calculation.

In Jia, Yue, and Mark Harman presented a comprehensive analysis, survey and shows the results of several

development trend analyses of Mutation Testing. These analyses provide evidence that Mutation Testing techniques, process, operators and tools are reaching a state of maturity and applicability [1].

The basic idea of this testing technique is, a program that is covered by unit tests, and the test cases are verified, and all tests have been passed for the given program, then that program is ready to apply for the mutation testing.

Mutation testing is like small changes or replace some part in the main program. For example, "= =" can become "! =" while "<" can become ">=". In other case, complex mutations mean rework the order to execute of code or may remove some lines of code [7]. Table 1. Shows a simple example of original code and corresponding possible mutant code.

Table 1. Simple example to mutant code

Original Code	Sample changes	Mutant Code
int large(int s, int p)		int large(int s, int p)
{		{
if(s==p)	1	if(s != p)
return s;	2	if(s <= p)
	3	if(s >= p)
	4	return p;
else		else
Return p	5	Return s

Typical mutations are known as replacing an addition with a subtraction, negating or short cutting conditions, changing the values of constants and literals, commenting out a line and many more [8]. Apply one mutation operator in the program, this small change in main program is known as mutant program

2.2 Mutation Score

Once the code has been mutated, it is tested by different test cases. If the mutants are executed or detected by at least by one or more test cases then it said to be killed, but mutants which generates similar result as the original program then it cannot be destroyed or killed, and it is said equivalent or live mutant. Sometimes test cases are unable to find the differences between original program and live mutants, because they produce the same output. Our objective is to kill as many mutants as possible so to do so additional test cases are added to kill these live mutants. If the new test cases have the ability to find out the changes then it can kill these mutants, and then these mutants are dead mutants [9].

Pawar Sujata G. and Idate Sonali R, their research objective is: Different types of mutation operators are explained for C# and using mutation operator test cases can be generated from execution trace [9].

Based on mutants live or dead mutation score is calculated, which indicates the quality of the input test suits. Mutation adequacy score (MAS) is calculated by the equation (1), which is defined as the percentage of dead mutants with the total number of mutants. [1,4].

$$MAS = (Kn / M) * 100 \quad (1)$$

Where, MAS, Kn, and M are mutation score, number of killed mutants, and total number of mutants respectively.

2.3 Mutation Process

The process of mutation testing is described in Fig. 1. A program P and a set of mutant P' is generated for mutation testing. Small changes or replace some specific portion of the original program is called mutants. Next, a test set T is applied to test the program. But before the mutation analysis, test sets are successfully executed against the original program p. If P is correct, set of test cases T will run for each live mutant P'. If the test cases successfully detect the mutant, then it will say to killed or dead mutant, if it can't then it is live or survived mutant otherwise program p may have errors.

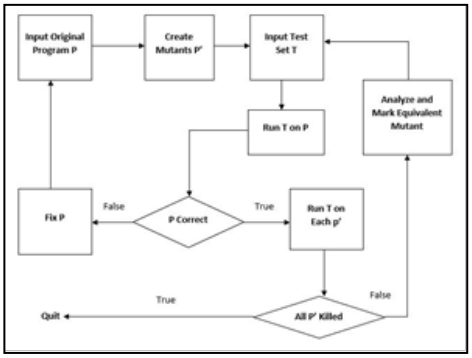

Fig. 1. Mutation Testing Process [1, 10]

Test cases may not kill all mutants, so there might be some surviving mutants. To kill these surviving mutants, program tester needs to improve the test set T by adding more test cases. However, test sets can't kill all mutants, because some mutants give the same output again and again. They are functionally equivalent but syntactically different to the original program. It is impossible to finding all equivalent mutant automatically because of undecidable program equivalence [11,12].

In [11,12], shows that excessive test sets not capable to kill all mutants because some mutants produce same output. Though some mutants are syntactically different from the original program but functionally equivalent. It is impossible to finding all equivalent mutant automatically because of undecidable program equivalence. They present a technique that uses mathematical constraints, originally developed for test data generation, to automatically detect some equivalent mutants and infeasible paths.

In [10] an experimentation advances, outlines a design of mutation testing process and envision a system to which a programmer can submit a program unit and get back a set of input/output pairs that are guaranteed to form an effective test of the unit by being close to mutation adequate.

Finally, the mutant adequacy score known as mutation score is calculated using the formula no. 1. Final objective of this mutation process is to improve this mutation score, by adding more test cases iteratively. Higher mutation score means the quality of the test cases of the program.

3. Analysis of different Mutation Testing Tools

There are so many testing tools for different platform, such as, Mujava [13], Muclipse [14], Jester [15] and Judy [16] for Java language, MutaTesting [17], MutaGenesis [18,4] for PHP language and so on. For C Sharp (C#) platform Visual Mutator [19], Cream [20] [21], Ninja Turtles [22] and Nester [23] is mostly use for mutation testing and free to use. NUnit [24], NCover [25], MSTestFinally[26], these tools are also needed for testing.

3.1 Visual Mutator

Visual Mutator version 2.1, is an open source mutation testing tool developed by Piotr Trzpil, runs as a visual studio extension type tools, but also can be accessed from command line to generate mutants for the testing program. This tool provides GUI with operator viewer, mutant viewer, mutation score, killed mutant, live mutant, and used test cases. GUI support to view and select the class or methods from the original program, test cases, standard and object operators. GUI also provides us to view the mutant operators and mutant codes. Some features of this tool are given below:

- Compilation the mutating code in Common Intermediate Language.
- By using build in and custom mutation, operators can create categorical mutants.
- After any modification it is able to check code fragment in C# and IL languages.
- Ability to view modified code fragments in C# and IL languages.
- Support Nunit and XUnit. Also run Unit test but not for single method.
- Have a good user interface and able to check the details about any mutant right after the start of the mutation testing process.
- Can view the mutation score and passed or failed tests information.
- Can save the result in an XML file.

3.2 Cream

Cream version 3.0, is an open source mutation testing and external application tool. This tool provides operator viewer, mutant viewer, mutation score, killing mutant, live mutant and support SVN client. It supports standard and object operators and .NET Framework 4.0, visual studio 2008 or higher version, NUnit[25], NCover[26], MSTest[27], Microsoft Excel 2007 or higher version. It also provides a well documentation with details to see the result of the whole mutation process in Microsoft excel.

3.3 Ninja Turtles

Ninja Turtles version 0.8.1.1, is an open source mutation testing and command line tool. It supports standard operator, MSTest and NUnit Version 2.6.4 and the project must be in .NET Framework 4.5.1 or higher. However, ninja turtles don't have proper documentation that how to operate and create mutant. If any user need to use this then user must go, throw the source code which is available in the code plex website [22]

3.4 Nester

Nester version 0.3 alpha is also an open source mutation testing tool, which followed the Jester mutation testing tool [15] for Java. It supports only standard operators, NUnit framework version 2.4.2 preferable, visual studio 2005 IDE, and .NET framework 2.0.

3.5 Mutation Testing Tools Summary

Visual Mutator and Ninja Turtles use commend line where Cream and Nester are external application to use mutation testing. But Visual Mutator have extra benefits that it is an extension of visual studio. With Visual Mutator and Cream have some facilities like Operator selection, Mutant Viewer, Mutation Score Calculation where Ninja Turtles and Nester don't have this facility. Most important part is visual mutator and cream contains standard & object mutators where Ninja Tutles and Nester contains only Standard. To test mutation testing all tools need NUnit testing but have some version specification. Visual Mutator, Cream can use MSTest. Table 2 shows the similarities and differences of those four tools, that are discussed above.

Table 2: Mutation Testing Tools Comparison

	Visual Mutator	Cream	Ninja Turtles	Nester
Interface	Visual studio extension and command line	External application	Command line	External application
Operator selection facility	Yes	Yes	No	No
Mutant viewer	Yes	Yes	No	No
Mutation score calculation	Yes	Yes	No	No
Operator type	Standard & object	Standard & object	Standard	Standard
NUnit Test	Yes	Yes	Yes (V 2.6.4)	Yes (V2.4.2)
MSTest	Yes	Yes	No	No
.NET Framework	All	4.0 +	4.5.1	2.0

Different mutation testing tools generate different type of mutants based on mutation operators. There are two types of operator's standard and object operators. Table 3 shows the list of mutant operators with short description.

Table 3: Mutation operators: standard and object

Operator Type		Abbreviation	Name
Standard	Common	AOR	Arithmetic operator replacement
		LCR	Logical connector replacement
		LOR	Logical operator replacement
		ROR	Relational operator replacement
	Uncommon	SOR	Shift operator replacement
		OODL	Operator deletion
		SSDL	Statement block deletion
		ABS	Absolute value insertion
		ASR	Assignment operator replacement
		UOI	Unary operator insertion
		UOR	Unary operator replacement
Object	Common	DMC	Delegated method change
		EHR	Exception handler removal
		EXS	Exception swallowing
		ISD/ISK	Super/base keyword deletion
		JID	Member variable initialization deletion
		PRV	Reference assignment with other compatible type
	Uncommon	DEH	Method delegated for event handling change
		EMM/EAM	Accessors, modifier method change
		EHC	Exception handling change
		MCI	Member call from another inherited class
		EOA	Reference assignment and content assignment replacement
		EOC	Reference comparison and content comparison replacement
		IHD	Hiding variable deletion
		IHI	Hiding variable insertion
		IOD	Overriding method deletion
		IOK	Overriding method substitution
		IOP	Overriding method calling position change
		IPC	Explicit call of a parent`s constructor deletion
		OAO	Argument order change
		OMR	Overloading method contents change
		PRM	Reference assignment with other compatible type

4. Experimental Analysis and Results

To perform the experiment of these four mutation testing tools, we developed a program that consist of 785 lines of code. To test this program 98 test cases are developed by NUnit and MSTest testing tools. Mutation score is calculated after testing the program with all these four mutation testing tools. Results are analyzed from standard and object operators' perspectives.

Table 4 and Table 5 shows mutation scores of different tools based on common and uncommon standard operators, and object operators.

Table 4: Mutation Score of Standard Operators

					Common Operators						Uncommon Operators				
	Mutation Score	Total Mutant	Killed Mutant	Live Mutant	AOR	LCR	LOR	ROR	SOR	OODL	SSDK	ABS	ASR	UOI	UOR
Visual Mutator	85%	301	256	45	48	x	x	100	x	100	53	x	x	x	x
Cream	47.71%	371	177	194	24	19	x	60	x	x	x	29	76	160	3
Ninja Turtles	75%	1076	816	260											
Nester	66%	135	90	43											

Table 5: Mutation Score of Object Operators

					Common Operators							Uncommon Operators						
	Mutation Score	Total Mutant	Killed Mutant	Live Mutant	DMC	HER	EXS	JID	JTD	PRV	DEH	EMM	EAM	ISD	MCI	EOA	EOC	IHD/IHI/IOD/IOK/IOP/IPC/ISK/OAO/OMR/PRM
Visual Mutator	57%	224	129	95	2	x	x	4	4	30	x	106	44	x	34	x	x	x
Cream	22.81%	57	13	44	x	x	x	8	x	9	x	x	x	x	x	x	40	x
Ninja Turtles	x	x	x	x														
Nester	x	x	x	x														

Ninja Turtles and Nester doesn't provide the statistics of mutant generation list based on operators, but Visual Mutators and Cream provide the details.

From Table 4 and Table 5 it is obvious that among these four tools, Visual Mutator's mutation score is the best for both standard and object operators. Ninja Turtle and Nester don't generate mutants for object operators. Though Ninja Turtles generates mutants only for standard operators, but its mutation score is second best.

5. Conclusion

Though mutation testing is not highly applied in software industry, but it is one of the strongest testing techniques to verify the effectiveness of the test cases. In mutation testing, one of the major drawbacks is high implementation cost. However, selecting a good quality tools may reduce the cost.

Developing a purely error free software is expensive, and in some extent impossible task. But by choosing the best tool we could reduce errors in a significant number as well as cost. In this study, we discovered that Visual Mutator is the best among these four tools, Visual Mutator, Cream, Ninja Turtles, and Nester. This study also reveals why different mutation testing tools generates different mutation scores.

References

[1] Jia, Yue, and Mark Harman. "An analysis and survey of the development of mutation testing." IEEE transactions on software engineering 37.5 (2011): 649-678.
[2] Papadakis, Mike, Nicos Malevris, and Marinos Kintis. "Mutation Testing Strategies-A Collateral Approach." ICSOFT (2). 2010.
[3] Zhang, Lingming, et al. "Regression mutation testing." Proceedings of the 2012 International Symposium on Software Testing and Analysis. ACM, 2012.
[4] Guru99. (2009, 20 jan). "Learn Mutation testing" Retrieved from http://www.guru99.com/mutation-testing.html.
[5] Rani, Shweta, Bharti Suri, and Sunil Kumar Khatri. "Experimental comparison of automated mutation testing tools for java." Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on. IEEE, 2015.
[6] R. Lipton, "Fault Diagnosis of Computer Programs," Student Report, Carnegie Mellon University, 1971.

[7] Jeremy Jarrell. (2010, 17 June). Retrieved from www.simple-talk.com/dotnet/.net-tools/mutation-testing/
[8] Filip Van Laenen (2012, April). Retrieved from www.accu.org/index.php/journals/1929
[9] Pawar Sujata G. and Idate Sonali R., "Mutation Testing Using Mutation Operators for C# Programs" International Journal of Advance Research In Computer Science and Software Engineering, Research Paper Vol.4.7, July 2014.
[10] J. Offutt and R. H. Untch, "Mutation 2000: Uniting the Orthogonal," In Proceedings of the 1st Workshop on Mutation Analysis (MUTA-TION'00) , published in book form, as Mutation Testing for the New Century. San Jose, California, 6-7 October 2001, pp. 34–44
[11] T. A. Budd and D. Angluin, "Two Notions of Correctness and Their Relation to Testing," Acta Informatica, vol. 18, no. 1, pp. 31–45, March 1982.
[12] A. J. Offutt and J. Pan, "Automatically Detecting Equivalent Mutants and Infeasible Paths," Software Testing, Verification and Reliability, vol. 7, no. 3, pp. 165–192, September 1997.
[13] J. Offut and N Li. (2011) µJava Homepage https://cs.gmu.edu/~offutt/mujava/.
[14] B. H. Smith and L. Williams, "An Empirical Evaluation of the Mujava Mutation Operator" in Testing Academic and Industrial Conference Practice and Research Techniques-MUTATION, 07, 2007, pp. 193-202.
[15] I. Moore, "Jester-a Junit test tester," in 2nd International Conference on Extreme Programming and Flexible Processsed in Software Engineering, Italy, 2001.
[16] L. Madeyski and N. Radyk, "Judy a mutation testing tool for Java," IET Software, vol. 4, no. 1, pp. 32-42, 2010.
[17] GitHub Retrieved from https://github.com/Halleck45/MutaTesting.
[18] GitHub Retrieved from https://github.com/padraic/mutagenesis.
[19] VisualMutator Website,https://visualmutator.github.io/web/.
[20] CREAM Website,www.galera.ii.pw.edu.pl/~adr/CREAM
[21] Derezińska, Anna, and Anna Szustek. "CREAM- A System for Object-oriented Mutation of C# Programs." Annals Gdansk University of Technology Faculty of ETI 5 (2007): 389-406.
[22] Ninja Turtles Website https://ninjaturtles.codeplex.com/.
[23] Nester Website, www.nester.sourceforge.net/.
[24] Nunit Website, www.nunit.org/.
[25] NCover Website https://www.ncover.com/
[26] Microsoft website: https://www.microsoft.com/en-us/
[27] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on Test Data Selection: Help for the Practicing Programmer," Computer, vol. 11, no. 4, pp. 34–41, April 1978.
[28] R. G. Hamlet, "Testing Programs with the Aid of a Compiler," IEEE Transactions on Software Engineering, vol. 3, no. 4, pp. 279–290, July 1977.
[29] Dong Jun,Xu Miao,Pan Yun-he,Ye. Statistic Model-Based Simulation on Calligraphy Creation [J]. Chinese Journal of Computers, 2008, 31(7): 7720-7725

Authors' Profiles

Tariful Alam: Post-graduated in computer science from American International University-Bangladesh, major in Software Engineering.

A. G. M. Zaman: Assistant Professor, Department of Computer Science in American International University-Bangladesh (AIUB). His research interest in the field of Data Mining, Machine Learning, Algorithms.

How to cite this paper: Tariful Alam, A. G. M. Zaman. " Experimental Comparison of Mutation Testing Tools for C Sharp Language ", International Journal of Education and Management Engineering (IJEME), Vol.10, No.5, pp.28-34, 2020. DOI: 10.5815/ijeme.2020.05.04