Static Program Slicing- An Efficient Approach for Prioritization of Test Cases for Regression Testing

Jyoti Arora
Rukmini Devi Institute of Advanced Studies
2A & 2B, Phase-1,
Madhuban Chowk, Outer Ring Road,
Rohini, Delhi-110085

ABSTRACT
Regression Testing is performed after modification of the program or software; it classified the existing test cases into re-used test cases and affected test cases after modification of the code. Test case prioritization is an approach of arranging the existing test cases in manner that most affected test cases (that generated maximum number of faults) test first after the other one. There are many techniques used for prioritization test cases at the time of regression testing. This paper present a new approach for prioritization of test cases using static executable program slices for regression testing. Program slicing is a process to classify the program into number of parts based on various types of dependencies between program statements. This paper presents an overview of basic concept of generating static program slices and on the basis of these programs slices prioritization of test cases at the time of regression testing

Keywords
Regression testing, Program dependency, static program slices, test cases prioritization and execution history

1. INTRODUCTION
Regression Testing is performed after modification of program code/software. After modification of program, it is very difficult to find out affected test cases due to modification of program. Regression Testing is a technique that classified existing test cases into two parts, re-used test cases after modification and affected test cases after modification. Test case Prioritization is a technique of re-arranging the existing test cases in a manner that test cases that affected more are executed first that improve the testing quality[6]. Many techniques used by researchers for prioritization of test cases, this paper presents an executable static program slicing technique for test cases prioritization at the time of regression testing. The concept of Program Slicing is to decompose the program into small units depends on various types of dependencies (data dependency, control dependency, call dependency and so on) between the program statements. Program slicing is applied into number of application for example in computer understanding, debugging, and testing and program comprehension by slicing the program into smaller parts. This paper shows the process of automatically generation of static program slices from the program and used these static program slices for prioritizing test cases at the time of regression testing.

2. PROGRAM SLICING
In the software development life cycle software testing plays very import role, with the help of software testing compare the estimated and actual result of software by executing a program or system with the intent of finding different types of faults. There are two approaches for performing software testing one is functional testing and other one is structural testing. In the case functional testing, only focus on functional part of the program and ignore internal details while comparing estimated and actual result. In the case of structural testing, focus on internal structure of the program while comparing estimated and actual result and finding out faults. The process of structural testing is only focus on Software Testing is the process of evaluating a system by comparing its actual and expected result manually or automatically.

Complete structural testing is a time consuming task and not possible. Sometimes, for many properties, only a small portion of the program is relevant. This can be done with the help of slicing.

Slicing is an important testing technique, it helps in understanding of the program or software by decompose the program into smaller part depending on the different types of dependencies (data, control, method call etc) between the statements. With the help of program slicing, each slice only containing statement that relevant to specific variable and ignore other statements.

Program slicing approach can be classified depending upon the run-time environment and slicing direction. Depending upon the run time environment, slicing can be slice or dynamic and depending upon the slicing direction, slicing can be forward or backward slicing.

For example figure 1. Shows C' language code.

```
1 .void main()
2. {
3.  inti,sum,mul;
4.  i=1;
5.  sum=0;
6.  mul=1;
7.  while(i<=10)
8.  {
9.  sum=sum+i;
10. mul=mul*i;
11. i=i+1;
12. }
13. printf("%d",i);
14. printf("%d",mul);
15. printf("%d",sum);
16. getch();
17. }
```

Figure 1. Sample c' program code

Figure-2 shows the demonstration for various program slices of program1. Figure 2. Shows executable program slices of all variables of program 1, every program slice includes statements from program-1 that directly or indirectly affects the value of that program slice variable.

With the help of program slices, we decompose our program 1 into number of slices containing filtered statements that affected by the slice variable. Finally for slice S1, count the value of i variable, for slice S2, compute the multiplication function and for slice S3 compute the sum function.

S1(i, 13)	**S2(mul,14)**	**S3(sum,15)**

void main()
{
inti,sum,mul;
i=1;
while(i<=10)
{
i=i+1;
}
Printf("%d",i);
getch();
}
	void main()
	{
	inti,sum,mul;
	mul=1;
	i=1;
	while(i<=10)
	{
	mul=mul*i;
	i=i+1;
	}
	printf("%d",mul);
	getch();
	}
		void main()
		{
		inti,sum,mul;
		sum=0;
		i=1;
		while(i<=10)
		{
		sum=sum+I;
		i=i+1;
		}
		printf("%d",sum);
		getch();
		}

Figure 2. Executable Program Slices for above C'

3. PROGRAM SLICING PROCESS

The step by step activities for generating static program slicing are

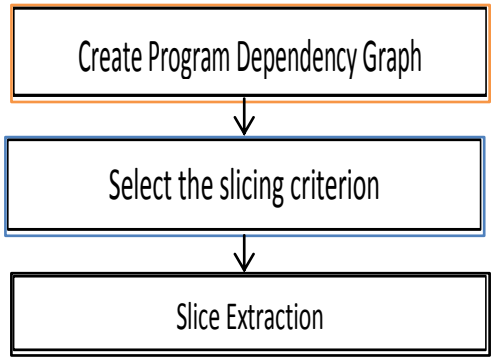

Figure3. Static program slicing process

1. **Select the slicing criterion**:- The slicing criterion specifies a point of interest (a statement in the program to be sliced) and interested variable. [2]
2. **Create Program Dependency Graph**:- The second step for program slicing process is to create program dependency graph on the basis of various types of dependencies between the statements[4].
3. **Slice Extraction**: - After creating dependency graph, the next step is to extract slices from program dependency graph using forward or backward approach.[7][12]

4. EXISTING PROGRAM SLICING TOOLS

Table 1 Comparison between Static Program Slicing Tools

	Wisconsin Tool	Unravel Tool	Kaveri Tool
PURPOSE	A Slicing tool evaluates C program	A Slicing tool evaluates ANSI C code	A Slicing tool evaluates Java code
APPLICATION	Debugging	Debugging	Program Comprehension
FEATURES	Slice C' Programs and highlighted.	Slice ANSI C' Programs and highlighted.	Slice Java Programs and highlighted.
	Program Dependency Tracking	Program Dependency Tracking	Program Dependency Tracking

5. PROPOSED APPROACH

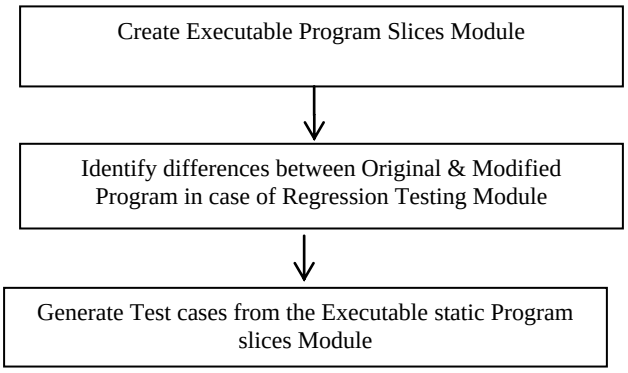

Figure 4. Proposed Approach

1. Create executable Static Program Slices Module
This module includes four steps:-

1. Select the Slicing Criteria.
2. Create Software Dependence Graph.
3. Slicing Extraction
4. Slicing Execution

In the first step of this module, input the slicing criteria from user –slice variable and slicing point.

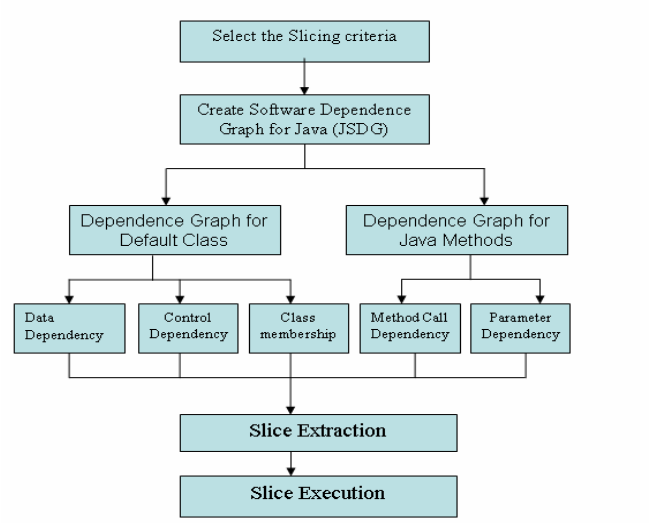

Figure 5. Step by step procedure for creation of executable program slices

In the second step of this module, create program dependence graph.

1. **Generate Test Cases from the Executable Static Program Slices Module**:- The executable static program slices contained line numbers of original program. On the basis of line numbers of individual executable static program slices generate test cases and maintained its execution history.
2. **Identify differences between Original & Modified Program in case of Regression Testing Module**:- This module includes four steps

1. Input original and modified java program.
2. Creation of flow graph of original and modified program.
3. Identify differences between flow graph of original and modified java program.
4. Display difference- new lines, deleted lines and modified lines.[12]

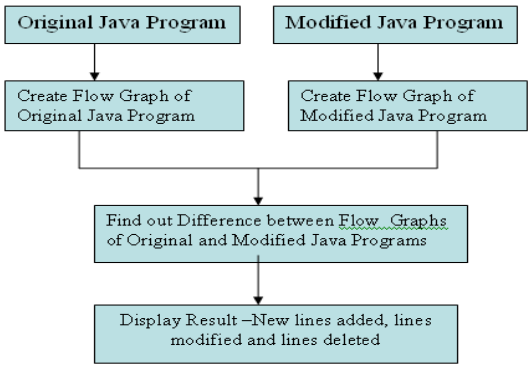

Figure 6. Step by step procedure for identify difference between original and modified program

3. **Prioritization of Test Cases in case of Regression Testing Module**:-This module includes two steps:-

1. **Search Modified lines into all slices of original java program & find out effected program slices**: - In this

step if slice contain modified lines, then it declared effect program slice.

2. **Compare execution history of original test suite with lines numbers of every effected program slices**:- In this step compare execution history of original test suite with lines numbers of every effected program slices and if both are equals then it declared effected test cases after modification.

3. **Prioritization of Test Cases** :- On the basis of maximum number of matches from the execution history of original test suite with lines numbers of every effected program slices, prioritized the test cases.[12]

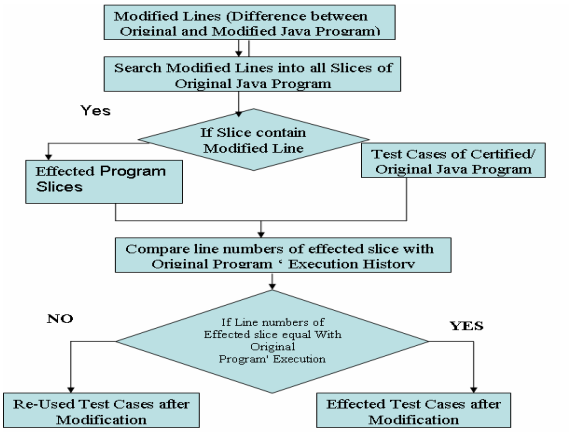

Figure 7. Step by step procedure for prioritization of test case in case of regression testing

6. DEMONSTRATE RESULT

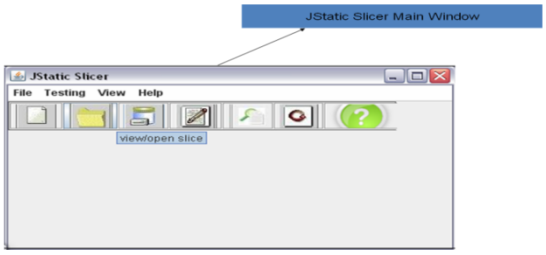

Figure 8. Main Window

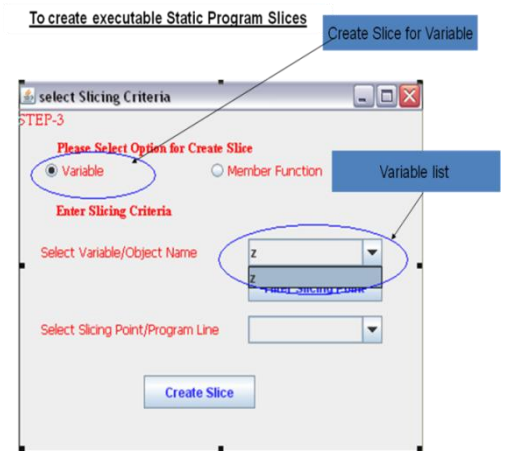

Figure 9. Creation of executable static program slices

Figure 10. Executed static program slice

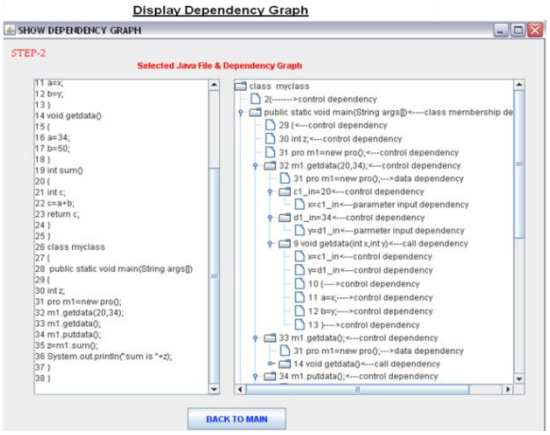

Figure 11. Dependency Graph

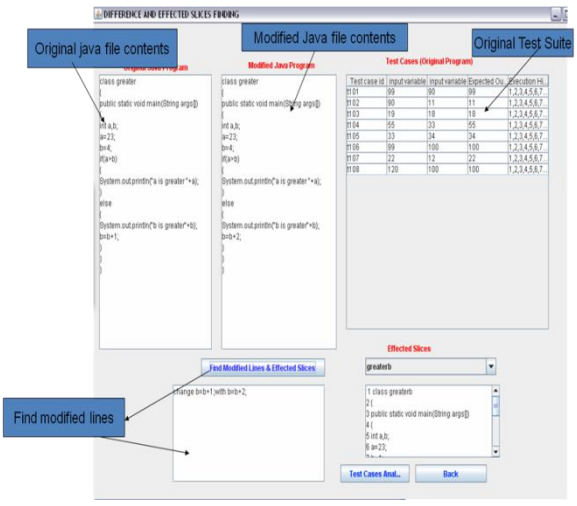

Figure 12. Difference and effected slices finding

Figure 13. Test case Analysis

7. COMPARISON WITH EXISTING STATIC SLICING TOOL

Table 2 Comparison with existing static slicing tools

	Wisconsin Tool	Unravel Tool	Kaveri Tool	Proposed Approach
Purpose	Evaluates C program code	Evaluates ANSI C program code	Evaluates Java Program code	Evaluates Java Program code
Application	Debugging	Debugging	Program Comprehension	Regression testing & program comprehension
Features	1.Slice C' program and highlights 2.Program Dependency tracking	1.Slice ANSI C' Programs and highlighted. 2.Program dependency tracking	1.Slice Java programs and highlighted. 2.Dependency Tracking	1.Create Executable static program slices. 2.Program dependency tracking 3.Dependency graph of complete java program

8. EXPERIMENTATION AND ANALYSIS

Let's suppose following are the existing test suite before modification of code/program. (we only included test case id and execution history in the existing test suite for our experimentation.

Table 3.Existing test suite of original program

Test case id	Execution history
t1	1,2,3,5,7
t2	1,2,3,5,7,8
t3	1,2,4,8,9
t4	1,2,3,5,5,6,7
t5	8,9,10
t6	1,2,8,9
t7	1,2,3,9,10

And modified lines are 3,4 and 5

Table 4: Test cases that includes modified lines

Test case id	Execution History
t1	1,2,3,5,7
t2	1,2,3,5,7,8
t3	1,2,4,8,9
t4	1,2,3,4,5,6,7
t5	1,2,3,9,10

According to executable static program Slicing approach affected test cases After modification are:-

Table 5. Number of faults and fault time associated with each test case of each test case

	t1	t2	t3	t4	t5
F1(for modified line no.3)	*	*		*	*
F2(for modified line no. 4)			*	*	
F3 (for modified line no. 5)	*	*		*	
No. of faults	2	2	1	2	1
Time	5	7	2	1	3

VTi=fault/time(rate of fault detection) [6]

The calculations are:

Vt1=2/5=0.4

Vt2=2/7=0.28

Vt3=1/2=0.5

Vt4=2/1=2

Vt7=1/3=0.33

Arrange the above Vti in deceasing order, since more the rate of fault detection more will be the priority.[6]

Hence the prioritized order is:T4,T1,T3,T2,T7.

In the above table

m=no. of faults = 10

n=no. of test cases = 10

So putting the values of m , n ,TFi(The position of

the first test in T that exposes fault i) in the

equation

APFD = 1 – TF1 + TF2+ ..... +TFm + 1

nm 2n[6]

Putting values:

APED=1-2+3+2 +1/10

5*3 =0.43

9. CONCLUSION & FUTURE SCOPE

Program slicing approach is used for various application i.e. program understanding, comprehensiveness, debugging, and testing using various types of dependencies between the instructions of the program. This paper shows process of automatic generation of static executable program slices and using these slices to find effected test cases after modification of the program. In this paper, try to working on simple java program to compute program slices where includes data, control, call, and method and parameter dependency, in future trying to work on other types of complex program including other dependencies between the program statements.

10. REFERENCES

[1] M.Weiser, "Programmers use slices when debugging",Communications of the ACM, Vol. 25, 1982, pp. 446452.

[2] Weiser, "Program Slicing", IEEE Transactions on Software Engineering 10(4), 352-357 (1984).

[3] F. Tip, "Survey of program slicing techniques", Journal of programming languages 3, 121-189 (1995).

[4] Z.Jianjun, "Applying Program Dependence Analysis to Java Software"In Proc. Workshop on Software Engineering and Database Systems,Pages 162-169, Taiwan, December 1998.

[5] L. Andrea De , "Program Slicing: Methods and Applications", IEEE international workshop,2001

[6] S.Praveen Ranjan, "Test Case Prioritization", Journal of Theoretical and Applied Information Technology, 2008

[7] P.Sandeep, "A New approach of program slicing" Mixed S-D (static & dynamic) slicing", IJARCCE, Vol.2, Issue 5, May 2013.

[8] C. Itti Hooda Rajender, "A Review: Study of test case generation techniques", IJCA, Vol. 107, Number 16, 2014

[9] E.Sebastian Elbaum, "Selecting a Cost-Effective Test Case Prioritization Technique", April 20, 2004

[10] S.Yogesh, K.Arvinder, "A new technique for version-specific test case selection and prioritization for regression testing," Journal of the CSI ,Vol. 36 No.4, pages 23-32, October-December 2006.

[11] [D. Gaurav, "Understanding regression testing techniques", IJCA, 2015

[12] J,Arora, "Generating & Prioritization of test cases Using static program slices" OSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, PP 37-44

[13] Indus Project. http://indus.projects.cis.ksu.edu/

[14] Unravel Project http://hissa.nist.gov/unravel/Wisconsin Program slicing project

[15] http://www.cs.wisc.edu/html/

[16] TopicsinProgramSlicing,http://www.cs.drexel.edu/~spiros/teaching/cs576/slides/5.slicing.pdf