IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 6, June 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

# A Survey on Program Slicing

[1]Rajlaxmi R ay [2] Subhasish Mohanty [3] Manoranjan Pradhan

[1, 2 ,3] Dept. of Computer Science & Engineering, Gandhi Institute for Technological Advancement(GITA), Bhubaneswar, Odisha, India, PIN-752054,.

## ABSTRACT

The program slicing is a method for automatically decomposing program by analyzing their data flow. Starting from a subset of a program's behavior, slicing reduces that program to a minimal form which still produces those behaviors. The reduced program called a "Slice" is an independent program guaranteed to represent faithfully the original program within the domain of the specified subset of behavior. Applying slicing technique to self were architecture can benefit S/W development in two main way. The first one concerns maintenance of a component based software. By using slicing tools on architectural description, we can determine which components might be affected when a given components is modified. Second architectural, reuse can be facilitated, while reuse of code is important, reuse of S/W design and patterns are expected to offer greater productivity benefits and reliability enhancements. This paper focuses on the various slicing techniques, like static slice, Dynamic slicing, simulation of dynamic slice, forward slicing, backward slicing, and functional slicing. The program slicing carried out using java which is an object oriented programming language.

***Keywords:***

*Program slicing, Program Dependency Graph ,Data Dependency Graph, Control flow Graph, Static slice , Dynamic slice, Backward slice, Forward slice, Amorphous slice, Application, Functional cohesion, Debugging.*

## 1. Introduction :

One of the program analysis techniques is program slicing. The application program slicing to continue within various software engineering activities such as program understanding, debugging, testing, program maintenance, complexity, measurement so on. It is a popular static analysis technique with a wide range of application for program comprehensive, software testing and software debugging. [5, 9, 6, 7,15]. It can also be used to interact the statements of a program that are relevant to given computation. The concept of program slice was introduce by wiser [1]. According to definition [2] the notation of slice was based on the deletion of statements [8] . A slice is on executable program statements that keeping safe the original behavior of the program with respect to subset of variables of interest and at a given program point. Feature of the programming languages such as procedures unstructured control flow, composite data types and pointers, and concurrency each require specific extensions of slicing is a pair <P,V>, where P is a program point of interest and V is the subset of the program variables. The slices mentioned so far are computed by gathering statement and control of predicates by way of a backward traversal of the program’s control flow Graph (CFG) or PDG, starting at the slicing criterion. If the values computed at the slicing criterion determine the fact is the statement under consideration is executed or not.

## 2. Proposed Work

### 2.1 Program Slicing

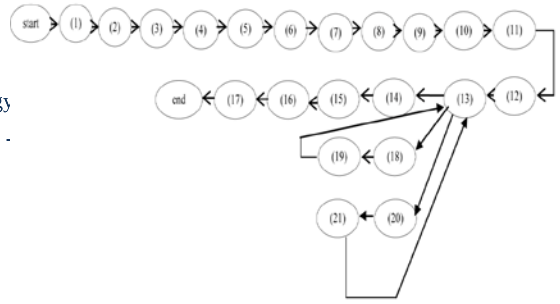
Program slicing is a technique which is program to resolve into the original element in to smaller parts after analyzing the data [2].

The application of program slicing included various software engineering activities such as program understanding, testing, debugging and maintenance. A program slicing depend upon those program statements which are related to the values computed at some program point. Suppose there is a program of binary search in which elements are in sorted position and an item is to be searched. This program dependent upon there conditions. Different techniques performed by slicing can be using by after making the program.

```
1.  int a [ 40 ] i, n, mid, high, low , item;
2.  int loc =0;
3.  printf ( “enter the number of element” );
4.  scanf (“% d”, & n);
5.  printf ( “enter the element in stored order” );
6.  for ( i = 0,i < n, i ++)
7.  scanf(“% d”, & a [ i ]);
8.  printf ( “enter the number to be searched” );
9.  scanf (“% d”, & item);
10. low = loc =0;
11. high=n-1;
12. while ( low + high )/2);
13. {mid=(( low + high )/2);
14. if (item= = a [mid])
15. printf (“search is successful”);
16. loc=mid;
17. printf(“\n loc items is %d” loc+1);}
18. else if (item < a [mid J])
19. high = mid2;
20. else
21. low= mid +1;}
```

### 2.2 PROGRAM DEPENDENCE GRAPH

Program Dependence Graph: - Program Dependence Graph (PDG) of a program P is the Graph G=(R,S). Here R is represents a



statement of the program P and S is represent inter statement data or control dependency edges in program P [1.5,9,19,24] The Program dependence graph contains two kinds of directed edges, which are data dependency edge and control dependence edge.

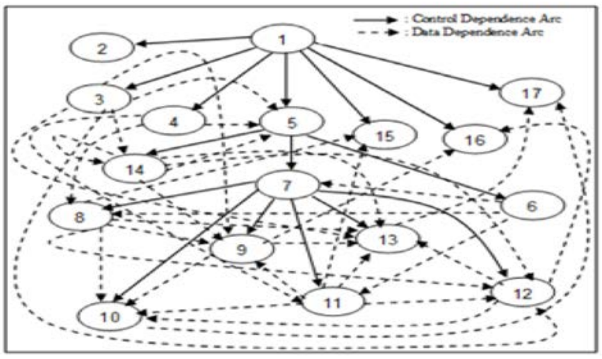


Fig.1 Program Dependence Graph

## 2.3 DATA DEPENDENCE GRAPH.

Program dependence Graph of program P and X and Y be two nodes in G [8, 30, 34]. Node is said to be data dependent on a node Y if there exist a variable Var of the program P such that the following hold.

1. The mode S defines Var.
2. The mode Y uses Var and
3. There exists a directed path form S to T which the Var is not defined.

Consider the given program in Fig.2 The node 5 has data dependence on node 3 and 4. If it because note 3 and 4 define var 'i' and 'n' individually and node 5 uses var I and n without redefinition. Node 7 is data dependent on node 6. It is because node 6 defines var j and node 7 uses var j and node 7 uses var j without redefinition.

### 2.3.1 CONTROL FLOW GRAPH

To give warning of program which can read only time for the input elements an search the item flow frequently. The Control flow Graph (CFG) is a data structure which make the Control flow Dependencies for each operation in a program explicit [3][9] Statement and components are exercised by a test if it is executed then the program will run on that test. Test data are sufficient criteria can be divided into three groups. Control flow based, data flow based, and program dependency (Graph) based.

Fig.2 Control Flow Graph (CFG)

### 2.3.2 TYPES OF PROGRAM SLICING

Slicing can be depending upon the run time environment.

1) Static slicing
2) Dynamic slicing
3) Backward slicing
4) Forward slicing.

#### 2.3.2.1 STATIC SLICING

According to wiser [1],[2],[3], a program slice consist of the part or component of a program thats effect the values computed at some point of interest referred to slicing criterion. Program slicing depend upon the parts or components of a program that effect the values which are referred to as a slicing criterion (5). A slicing a test depend upon a pair <X, Y> where "X" is the statement number and T is a variable. The constituent part which have a direct or indirect effect on the values computed at a slicing criterion <X, Y> are called the program slice with respect to the slicing criterion <X,Y>. Now static slicing is preformed on the total program static slicing simply means that the remove one variable and the eliminating all those condition which are using removing variable [14].

In weiser's approach, slices are computed by computing consecutive set of transitively is used for computing slices, only statically available hence this type of slice is static slice. A static slice is always some or greater than the dynamic slice. To calculate the static slice first construct the Program Dependency Graph of given a program. The calculate the static slice, with respect to a slicing criterion (PV) where P is a statement V is the variable used. In order to get a static slice for slicing a rule (PV) PDG nodes are traversed in backward reach ability using the concept of either breadth first search (BFS) or Depth first (DFS) starting from node P. Fig.3 shows a static slice of slicing rule (15,A) for the program given in Fig.1. The static slice with respect to a rule (15 A) consists of the nodes {1, 3, 4, 5, 6, 7, 8, 11, 14, 15} static slice WRT criterion (15-A)

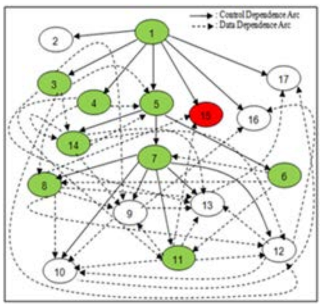


Fig.3 Static Slice W.R.T Criterion (15,A)

### 2.3.2.2 DYNAMIC SLICE

During program slicing rules contains the variable which produced on expected result on some input to the program. It takes the input supplied to the program during execution and the slice contends only the statement that caused the failure during the specific execution of interest. The advantages of dynamic slicing is the run time handling of arrays and pointer variables. Dynamic slicing will treat each element of an array individually, where as static slicing considers each definition or use of any array element as a definition or use of the entire array. Dynamic slicing usually smaller than static slices, thus allowing an easier localization of the bugs are more useful in program testing Dynamic slicing techniques. Compute ( praised ) exact value slices. In calculation of a dynamic slice, we analyze dependence from execution history. This record the execution of statements as the program executes. Dynamic slice is calculated from Program

Dependency Graph of (PGD) based on the decency analy is of the execution history. The statements that have not executed are removed from slice [12].

We calculate dynamic slice as follows:

1. All nodes in the program dependency graph are drawn dotted in the beginning.
2. The statement are executed in corresponding nodes in the graph are made to hard.

3. In order to get a dynamic slice Program Dependency Graph is traversed only for hard nodes in backward reach ability using the concept of either BFS or DFS starting from node P.
   All nodes reached during the traversal are made bold. The set of all bold node and are desired slice..
4. To think over carefully a particular execution of the program in fig.1 for the input value n=3,j=1 and j=3 and its corresponding dynamic slice in fig.4. The input value n=3,j=1and j=3 yields the execution history <1,2,3,14,6,7,11,12,13,14,52,62,72,8,9,10,142,53,15,16,17>. Now we explain the PDG of the program in fig.1 initially all the nodes in PDG are drawn dotted. To calculate the dynamic slice PDG is traversed based on the slicing criterion [12]. The dynamic slice with respect to criterion =(n=3,j=(i-3),15A) consist of the node the {1,3,4,5,6,7,8,11,14,15}.

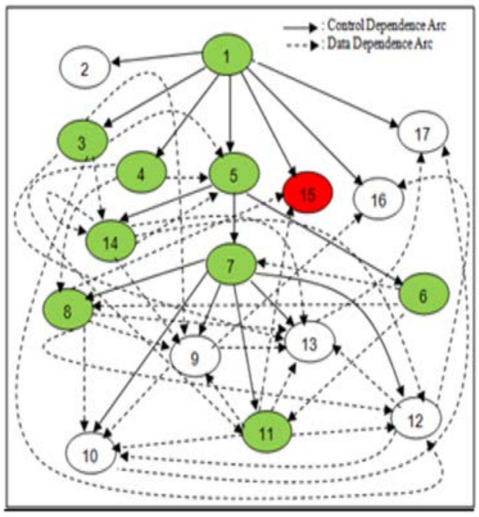


Fig.4. Dynamic Slice W.R.T Criterion (n=3, j=(1,3),15,A)

### 2.3.2.3 BACKWARD SLICE:

Backward slice to continue within all part of the program that might have not influenced the variable at the statement under executable dynamic slice of Program on slicing criterion C is any syntactically correct and executable program ‘P’ that is obtained from P by deleting zero or more statements, and when executed on program in put x produces an execution trace $T^x$ for which their exists the corresponding execution position q such that the value of Y in $T^x$ . The aim of dynamic slicing is find the slice with the minimal number of statement.

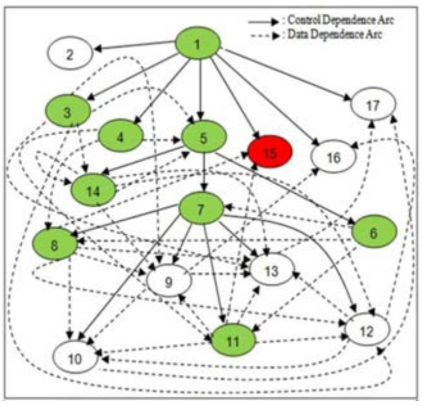


Fig.5 Backward Slice W.R.T Criterion (15, A)

### 2.3.2.4 FORWARD SLICE

Forward slice contains all the statements of P which might be influenced by the variable. The main application of forward slice are program debugging, program comprehension program analysis, software maintenance and testing [20,30]

To compute a forward slice from point P, compute forward reachability in the PDG from node P to using BFS or DFS.

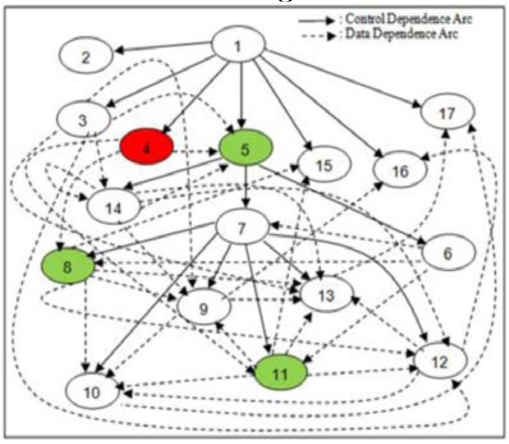


**Fig.6 Forward Slice W.R.T Criterion (4, n)**

### 2.3.2.5 AMORPHOUS SLICES:

Previously slicing techniques involve two requirements whereby the slice captures some projection of the original programs semantics and syntactic requirement where by the slice is constructed by deleting components from the original program. l Without slicing relaxas the syntactic requirement and thus allow additional trans formations to be applied [34]. The example below applied constant propagation and then traditional slicing, which can then remove the first statement.

```
a=42
b=a+2          b=42+2
```

### 2.3.2.6 APPLICATION

The application of the program slicing techniques dividing into branches in to a powerful set of tools for use in such divers application program understanding, verification automated computation of a several software engineering matrics, software maintenance and testing functional cohesion , dead code elimination, revers engineering parallelization of sequential programs, software portability, program integration so on. Also it can be been applied to many other problems areas. The following section lists come of the application of slicing in different fields.

## 2.4 FUNCTIONAL COHESION

Cohesion for the programmed is measured as on application of syntax preserving static program slicing. A Cohesive program is one in which the modularization of the program is performed correctly A cohesive function or procedure should performer the tasks that are related to each other. In slicing application in cohesion measurement application in cohesion measurement play & a vital role in object-oriented programming language. Chas ion them in the object oriented programming is based on encapsulation. An encapsulated code contains all the necessary data and number function associated with that object slice captures a thread through a program which is through a program me which is concerned with the compilation concerned with the compilation of some valuable. The idea of function cohesion us that a member function should perform related task different slice from a fuction, each for a different slice from a function, each for a different variable can be sliced let us consider a example for the action which are related. A function that performs division of two numbers calculated the quotient and reminder.

## 2.5 DEBUGGING:

Debugging is a generalize term which essentially means to step through a process in order to systematically eliminate errors. In programming Debugging is done when undesirable program execution or termination occurs. Debugging is the necessary process in almost any new software or hardware development process, whether a commercial product or an enterprise or personal application program. Dynamic slice in particular is appropriate when applying program slicing to debugging, as it products smaller slices and make available the inputs that caused the fault[35] . Debugging process includes various stages like defined below.

### 2.5.1 REVERSE ENGINEERING:

Reverse Engineering is the process of analyzing a technology specifically to determine how it was designed or how it operates. Reverse engineering concerns the problem to understand the current design of a program and the way this design differs from the original design.

#### 2.5.1.1 SOFT WARE MAINTENACE:

It is modification of a software product after delivery to correct faults, to improve performance or other attributes, or to accept the product to a modified environment. In software maintenance are to understand various dependencies in existing software and make changes to the existing software without introducing new bugs. [21].

### 2.5.2.2 TESTING:

Slicing helps to decompose which in testing, it makes test work faster and more efficient. The interrelated modules can be identified, which then can be tested separately without distributing the rest of the program. Because program slicing helps in understanding program by dividing it into slices, the task of testing can be allocated to a various testers. Each tester can test the slice in the program domain. [21].

## 3. CONCLUSION

The paper discuss about various type of slicing techniques like the program slicing, static slicing, dynamic slicing, backward slicing, forward slicing each slicing is related to C Language. In this paper the application of slicing in various areas like debugging, maintenance and re-engineering and testing are highlighted soft based purely.

## 4. REFERENCES

[1] Swarnendu Biswas and Rajib Mall, “Regression Test Selection Techniques: A Survey”, Information and Software Technology, Vol. 52, no. 1, January 2010.

[2] Jaiprakash T Lalchandani, R Mall, ”Regression Testing Based-on Slicing of Component-based Software Architectures”, ISEC ,vol. 79, no. 06, pp. 19-22, 2008.

[3] Rajiv Gupta, Mary Jean Harrold, Mary Lou Soffa, “An Approach to Regression Testing using Slicing”, ACM Transactions on Programming Languages and Systems, vol. 12, no. 1, pp. 26-60, January 1990.

[4] Hiralal Agrawal, Joseph R. Horgan, “Dynamic Program Slicing”, ACM SIGPLAN Notices, Vol. 25, No. 6, pp. 246-256, June 1990.

[5] M. Harman and S. Danicic, (1997), “Amorphous program slicing”, Proceedings of 5th International Workshop on Program Comprehension, Dearborn, Michigan, U.S.A., IEEE CS Press, pp. 70-79.

[6] J. Jiang, X. Zhou, and D.J. Robson, (1991), “Program slicing for C - the problems in implementation”, Proceedings of Conference on Software Maintenance, Sorrento, Italy, IEEE CS Press, , pp. 182-190.

[7] D. Binkley (1998), The application of program slicing to regression testing. Information and Software Technology, Special Issue on Program Slicing, 40(11-12):583 – 594.

[8] M. Kamkar(1993), Inter Procedural Dynamic Slicing with Applications to Debugging and Testing. PhD thesis, Linkoping University, Sweden. ISBN 91-7871-065-0.

[9] Raju Halder and Abortion Cortesi, Abstract program slicing on dependence

condition graphs, Science of Computer Programming, 2012.

[10]Torben Amtoft and Anindya Banerjee, A logic for information flow analysis with an application to forward slicing of simple imperative programs, Science of Computer Programming 64, 3–28, 2007.

[11] Liang D. and Harrold M. J. Slicing Objects Using System Dependence Graph. In Proceedings of the International Conference on Software Maintenance, IEEE, pages 358–367, November 1998.

[12] Varun Gupta, Jitender Kumar Chhabra, Dynamic cohesion measures for object-oriented software, Journal of Systems Architecture 57, 452–462, 2011.

[13] Tonella et al., Flow insensitive C++ pointers and polymorphism analysis and its application to slicing. In Proceedings of 19th International Conference on Software Engineering, pp. 433-443, 1997.

[14] Song and Huynh, Forward Dynamic Object-Oriented Program Slicing, Application Specific Systems and Software Engineering and Technology (ASSET'99). IEEE CS Press, 1999.

[15] G.B. Mund, R. Mall*, S. Sarkar, Computation of intraprocedural dynamic program slices, Information and Software Technology 45, 499–512, 2003.

[16] N.Sasirekha et al., Program Slicing Techniques And Its Applications , International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011.

[17] M. Weiser. Programmers use slices when debugging. Communications of the ACM, 25(7):446–452, 1982.

[18] M. Weiser. Program slicing. IEEE Transactions on Software Engineering, 10(4):352–357, 1984.

[19] Sonam Agarwal and Arun Prakash Agarwal, " Program Slicing using Test Cases", International Journal of Computer Applications (0975-8887), Volume 60 –No. 10 , December 2012.

[20] Automatic Distribution of Java Byte-Code Based on Dependence Analysis Roxana E. Diaconescu, Lei Wang, Michael Franz University of California, Irvine Department of Computer Science roxana, wangglei, franz @ics.uci.edu.

[21] On-line Trace Based Automatic Parallelization of Java Programson Multicore PlatformsYu Sun and Wei Zhang Department of ECE, Virginia Commonwealth University wzhang4@vcu.edu.