Automated Dynamic Slicing based UML Modeling for Phylogenetic Classification

Divanshi Priyadarshni Wangoo
CSE & IT Department
ITM University, Gurgaon
Haryana, India

ABSTRACT
This paper presents an efficient classification algorithm for categorizing evolutionary organisms using slicing techniques. Dynamic slicing excels in tracing out dependencies between executable statements. The nature of these dependencies aids in the determination of control statements in a program. Dynamic slicing technique imbibes the run time execution trace based on a slicing criterion. Dynamic slicing algorithms can trace both the backward and forward dependencies. The UML model is automatically generated from the source code to validate the forward and backward dynamic slicing algorithm. This paper shows the algorithmic implementation in NetBeans IDE 7.4. It provides a new platform for automated software engineering. The algorithm efficiently discovers the evolutionary relationship between organisms .Forward dynamic slicing algorithm helps in identifying the successors of the organisms and the backward dynamic slicing algorithm finds out the predecessors of the evolutionary organisms. Both the algorithms are based on dynamic slicing criterion at the run time execution trace. The integration of these phylogenetic algorithms deciphers the building complexity of the evolutionary organisms. It proves to have an advantageous classification encasement for jeopardized species.

General Terms
Dynamic slicing, Object-oriented.

Keywords
Forward Dynamic Slicing Algorithm (FDSA); Backward Dynamic Slicing Algorithm (BDSA); Phylogenetic System Dependence Graphs (PSDG); System Dependence Graph (SDG); Lines of Code (LOC); Software Development Lifecycle (SDLC).

1. INTRODUCTION
Slicing techniques and its applications have evolved over years through the ongoing researches in different domain. Program slicing has varied with all new technological programming relevant fields. The new forms of slicing techniques have emerged from the profound dependencies as a result of executable commands. Slicing was basically practiced for programming languages that admitted the interactions among objects. Object interactions involve the message passing scheme with which one object can communicate with another object. All the object-oriented features like classes, objects, inheritance, abstraction, polymorphism, and encapsulation can be featured through various dependence graphs based on slicing methodology [1]. The programming statements of an object oriented language features versatile control flows that direct the implementation of the executable program file. By recognizing the dependent relations in the program code, the complex inheritance constructs can be verified with ease. Also, other programming jobs are simplified which becomes difficult with the intent of increasing Lines of Code (LOC). Slicing proficiencies can develop phyletic taxonomy of species by tracing the historical hierarchy of evolutionary organisms. It is necessary for the species requiring categorization to have their morphed mutable features inheritable form their predecessors. Historic integrality has an immense hereditary patterned system that recognizes the development of succeeding genesis. Thus, to make the taxonomical evolution traceable to a greater extent, slicing has been incorporated to discover the next future generation species. A taxonomic group is any organism bearing natural relations. These relations by law of nature are governed by dependent relationships between the organisms in the hierarchy. The organism's existence comes from either the immediate predecessors or the older predecessors from which the immediate predecessor derives. The inheritance derivation is the framework for the ontogenetic species.

This paper presents an efficacious dynamic slicing algorithm for the classification of birds. The advantage of classification algorithm amounts to precise knowledge of predecessors and successors of a particular species. The dynamic slicing technique is preferred as the implementation of the algorithm can be tested at the execution run time. This verifies the validity of the algorithms for large data sets involving numerous inheritance hierarchies. The species of birds have evolved over time from different predecessors that all have the same ontogenetic base of aves species. The evolution bases its ground on the dependence of features that one species develop from their immediate predecessors. The immediate predecessors may inherit different characteristics from any number of predecessors. This governs the hierarchical dependence that exists with the visible or behavioral characteristics shown by the species. Keeping in view the dependencies among the taxonomical community of species, the slicing methodology can be employed for the given group. The main criteria used for categorizing the species by slicing is by tracing the inheritance hierarchy of a given organism and applying the slicing algorithms for its predecessor or successor determination. This paper presents two algorithms based on dynamic slicing that yields the successors or predecessors of an organism. The algorithms are called Forward Dynamic Slicing Algorithm (FDSA) for computing the successors and Backward Dynamic Slicing Algorithm (BDSA) for the computation of the predecessors of a given species. The execution trace upshots are given which show the validation of the FDSA and BDSA algorithms for different inputs. Thus, the application of dynamic slicing algorithms aid in the production of more effective classification scheme and reduces the LOC's of the programming code to an estimable track.

2. EVOLUTION IN ORGANISMS

Evolution is ontogenetic operation that relates one species nascence with time. It is an automatic process and is governed by various environmental ingredients. It gives a clear historical in-depth into genesis of organisms. The evolutionary relationships between organisms can be worked out with the implementation of the slicing algorithm. Slicing is an excellent procedural technique which channels the discovery of hidden out relationships. These shrouded relationships determine the dependable characteristics that one species at the hierarchical layer share with the predecessor species. The dependable features also reveal the future genetics that would imbibe in the progeny of the species. With these knowledge base structures, the immediate or super predecessors who have lead to the organic evolution of successors can be determined with the backward tracing of the evolutionary tree. Similarly the future genesis of the species is discovered by forward tracing of the organic tree. The species coexistence is driven by the predecessor foundation stack that related the species with their taxonomical group.

2.1 Phylogenesis- The Organic Evolution

The process of organic evolution is driven by mutable species which best adapt themselves to the inhospitable environmental surroundings. Darwin's theory explains the survival of the fittest species that have fought for the accommodation of their future genesis. From this comes the need for the phylogenetic tree construction that would govern the strongest features still existing and that have evolved over time with the mutation of species in progress. A phylogenetic tree is a visual representation of the relatedness of species by descent from a common ancestor. Evolutionary tree for flamingo birds is given in "Figure 1" below.

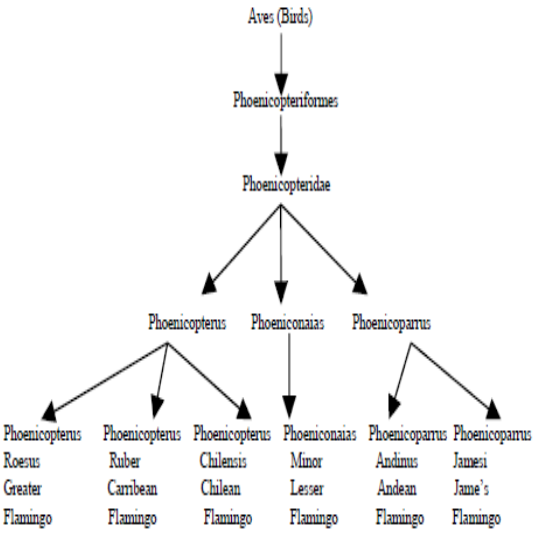


Fig. 1. Aves Classification

The classification tree of the above figure depicts the evolution of flamingo species from their ancestors. All the species of the birds derive from their grand ancestral Aves. The aves species through time and with mutation procedure gave birth to Phoenicopteriformes. Phoenicopteriformes is the immediate successor of Phoenicopteridae which are further classified as Phoenicopterus, Phoeniconaias and Phoenicoparrus. Phoenicopterus has three successor flamingoes named Great Flamingo, Carribean Flamingo and Chilean Flamingo. The Phoeniconaias species has only one successor as Lesser Flamingo and the Phoenicoparrus has two successor flamingoes named Andean Flamingo and Jame's Flamingo. The phylogenetic tree has five hierarchical levels and species at each level have either superior qualities than predecessors or lesser qualities than successors. This proves the evolution growth evolved at each stage with the development of the strongest features for the fittest survival. The flamingo species are jeopardized and their classification through slicing techniques will discover those features that govern the survival of the fittest strategy so that their endangered existence is protected.

2.2 Retracing Slicing technique

The concept of dependence graphs construction has evolved with current researches going on in distinguishable fields. The dynamic dependence relation that exists in the program is based on the slicing criterion supplied dynamically. The methods of dynamic slicing range from basic algorithm construction involving the dynamic flow concepts, dependence graphs to procedures for slicing [2]. Slicing techniques fundamentally covers the program debugging, understanding, maintenance, re-engineering and testing. Dynamic slicing is the best technique for localizing faults in the program when debugging is involved. Debugging process aims at finding, localizing, modifying and correcting errors by setting breakpoints [3].

Retracing slicing technique is a novel approach to slicing which will induce complacent existence of evolutionary relationship between organisms. It involves tracing the phylogenetic tree for a particular species in forward-moving and backward-moving mode to cognize the future and historical genesis. The retracing slicing technique is specifically designed to find out the dependable features that species share with their group. These dependable features are the decidability factor points governing the inheritable strength of the species. With the help of retracing dynamic slicing technique the possible predecessors and successors can be predicted at all the inheritance level. The technique is based on a dynamic slicing criterion at execution run time trace and results in the precise slice points which determine the exact number of successors and predecessors that a species consists of at the evolutionary hierarchy tree.

2.3 Evolutionary Data Sets for Flamingo Species

The data set for determining the historical evolution is taken for flamingo species. The data sets consist of the phylogenetic tree for flamingo species which is shown is Fig.1. The classification tree for aves (birds) trace the evolution in both the directions i.e., forward and backward way. The forward way of classification is used for finding successors and the backward way is used for the predecessor's calculation.

3. DYNAMIC SLICING TECHNIQUES AND ALGORITHMS

Slicing techniques work by building the dependence graphs to locate the dependent statements in a program. Various forms of slicing graphs have come into existence for dependencies occurring in different programming languages. Originally the dependence graphs for programs were constructed to determine the flow of controls in a program. Dependence graphs have been built for interprocedural programs that are generic in nature [4].

3.1 Phylogenetic System Dependence Graphs (PSDG)

Phylogenetic System Dependence Graph (PSDG) is a directed System Dependence Graph (SDG) whichaims at tracing the evolutionary dependencies between hierarchical statements in a program. The PSDG is a novel approach to the construction of dependence graphs in the phyletic sphere of species. The graph consists of nodes symbolizing the statements in the program and arcs represent the dependencies between the nodes. The directed solid lines represent program control dependencies, directed dashed lines represent the program data dependencies and the directed dashed dotted lines represent the parameter & procedural bindings occurring between the nodes of the PSDG. The shaded areas in the graph represent the procedural binding points which are represented through directed dashed dotted lines form the function calling nodes to the function definition nodes. The dashed lines connect the nodes which are dependent on the data for execution to the data dependent nodes in the program. The directed solid lines represent the normal control execution flow of the program. The PSDG's for successors and predecessors are shown in "Figure 2" and "Figure 3" respectively.

The conventions used in the PSDG's are same for successors and predecessors discovery. The node **PS** in the graph of Fig.3, represents the PSDG of " Figure 2," with the only difference in tracing execution lines for predecessors. The PSDG works by tracing the dependable flows in the program statements. The control flow determines the execution flow of the program at run time. The data dependent flows show those statements that depend on the data entered by the programmer for execution. The parameter and procedural bindings determine the binding of the procedure or function calling with the function definition and declaration. Thus, all the parametric variables in the statements build the PSDG's for the discovery of successors and predecessors. The PSDG is used as an input to the FDSA and BDSA algorithms for finding successors and predecessors respectively and are described next. The successors and predecessors PSDG's for aves classification program are shown in "Figure 2" and "Figure 3" respectively as below.

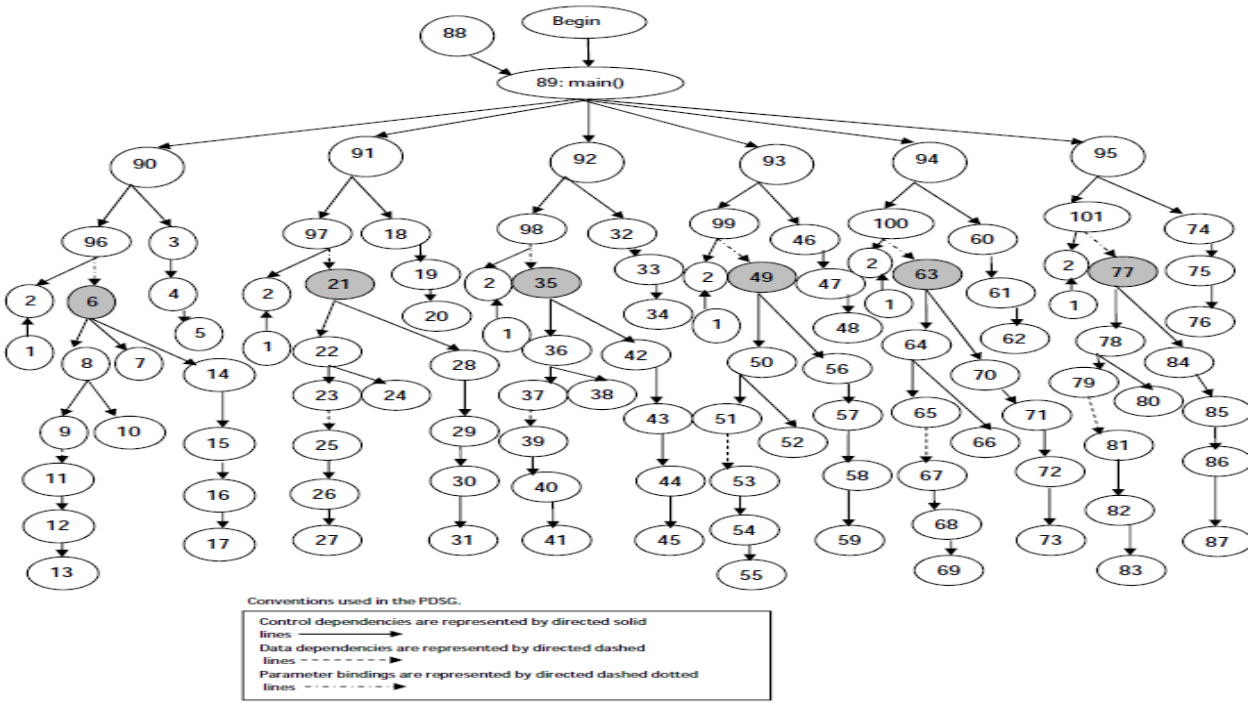


Fig. 2 PSDG for Successors program

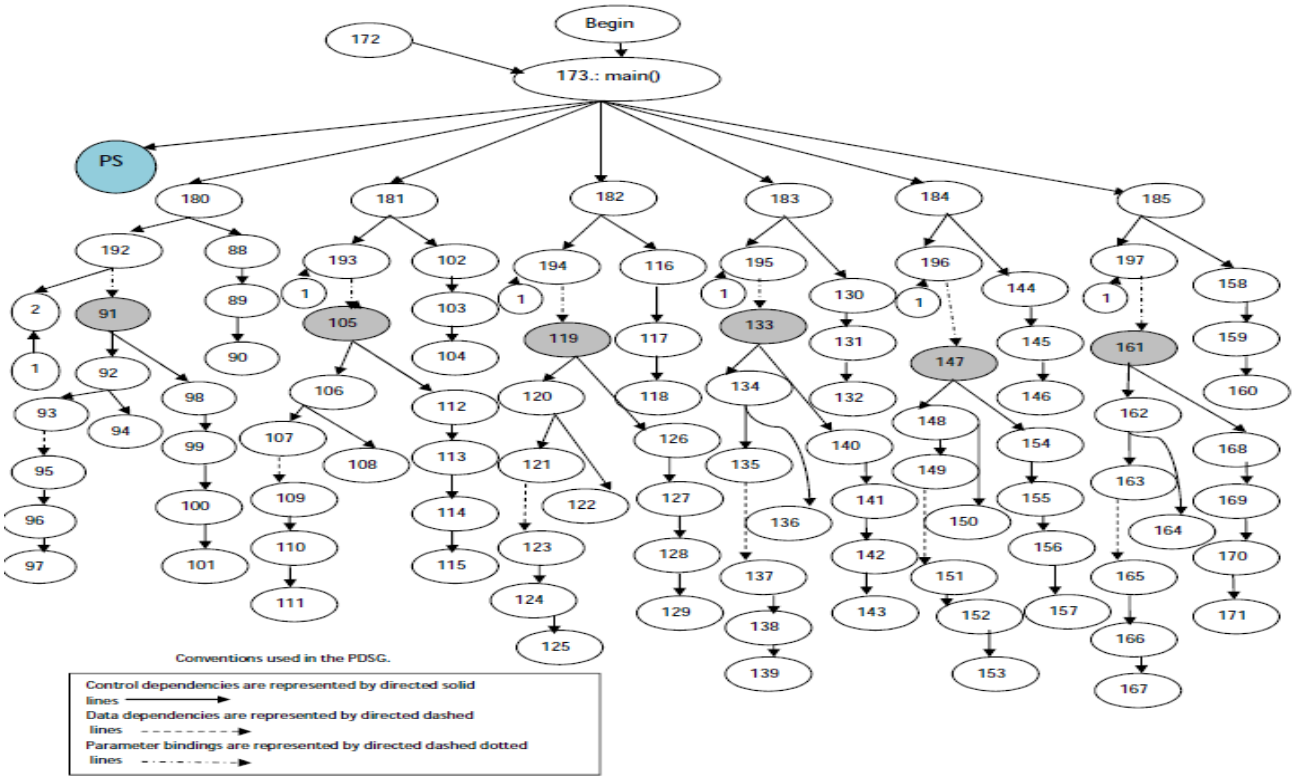


Fig.3 PSDG for Predecessors program

The above PSDG's are used as an input to the FDSA and BDSA algorithms for finding successors and predecessors respectively and are described in the following section.

3.2 Forward Dynamic Slicing Algorithm (FDSA)

The Forward Dynamic Slicing Algorithm (FDSA) is constructed for finding out the successors of the flamingo species. The algorithm takes as input the PSDG of "Fig. 2," and a dynamic slicing criterion. The output of the algorithm is the required successor of the species by tracing the forward dependencies in the program.

FDSA Algorithm:

Input: PSDG of "Figure 2," dynamic slicing criterion <N, v, f, e>, where N= Statement number in the program, v= the affected variable, f=flamingo species, e= dynamic execution trace. Su represents the successors of the flamingo species. C= Flamingo category.

Output: Successors of the flamingo species Su.

Procedure: getSuccessors()

Step1- Construct the PSDG for successors program.
Step 2- Repeat the following for the given phyletic tree
Step 3- For all f=0; f<=n ; f ++where n= total no of flamingo evolutionary species
{
Step 4- if (C==f)
{
Step 5- Track the forward dependencies form f
Step 6- Compute the forward slices with <N, v, f, e> at run time
Step 7- call getSuccessors(); method
Step 8- Print "Successors: + Su" if condition contained in if statement is true
}
Step 9- else { Print "No Successors" }
}
Step 10- Continue;
}
Step 11- Break;
}
Step 12- Stop

3.3 Backward Dynamic Slicing Algorithm (BDSA)

The Backward Dynamic Slicing Algorithm (FDSA) is constructed for finding out the predecessors of the flamingo species. The algorithm takes as input the PSDG of "Figure 3," and a dynamic slicing criterion. The output of the algorithm is the required predecessor of the species by tracing the backward dependencies in the program.

BDSA Algorithm:

Input: PSDG of "Figure 3," dynamic slicing criterion<N, v, f, e>, where N= Statement number in the program, v= the affected variable, f=flamingo species, e= dynamic execution trace. Pr represents the predecessors of the flamingo species.

Output: Predecessors of the flamingo species Pr.

Procedure: getPredecessors()

Step1- Construct the PSDG for predecessors program.
Step 2- Repeat the following for the given phyletic tree
Step 3- For all f=0 ;f<=n; f ++ where n=total no of flamingo evolutionary species
{
Step 4- if (C==f)
{
Step 5- Track the backward dependencies form f
Step 6- Compute the backward slices with <N, v, f, e> at run time
Step 7- call getPredecessors(); method
Step 8- Print "Predecessors + Pr" if condition contained in if statement is true
}
Step 9- else { Print "No Predecessors" }
}
Step 10- Continue;
}
Step 11- Break;
}
Step 12- Stop

4. TRACING THE ORGANISATIONAL TREE

Tracing the phylogenetic tree of flamingo species requires the computation of the forward and backward dynamic slicing criterion for successors and predecessors respectively. The slicing criterion is the decision breakpoint which will search the whole evolutionary tree in order to produce the output for the particular species in the hierarchy.

4.1 Thoroughbred of the Species

The flamingo species hierarchical inheritance exemplifies the throughbreding of the species in order to find the historical evolution. For this the Aves classification graph of "Figure 1," specifies the layers into the evolutionary inheritance. With this the immediate successors and predecessor of a particular species at a specific layer in the tree is noted and the algorithm is run keeping in view the result. For knowing the immediate successors of the species the FSDG is run on the specific node in the PSDG for successors. Similarly for calculation of immediate predecessors the BSDG is executed on the particular node in the PSDG for predecessors.

4.2 Code Implementation in Java

Java Source code for Successors & Predecessors is shown is and respectively. The Phylogenetic System Dependence Graph (PSDG) is constructed separately for forward tracing of successors and backward tracing of the predecessors. All the nodes in the PSDG of "Figure 2," and "Figure 3," correspond to the statement numbers in the programs in java respectively. The control flow, data flow and parameter or procedure binding dependencies can be traced from the statement numbers in the program and the nodes with their respective arcs of the PSDG.

4.3 PSDG Execution Flow for Successors and Predecessors

Automatic software development aims at improving the software quality and reducing the cost of production and maintenance of the software. Model transformations have been applied to slicing techniques particularly dynamic slicing for data accessing and slicing applications in various domains [5]. Integral UML models for class diagrams has been developed and used for the static structural and behavioral features of the architectural design of the software [6]. UML architectural designing constructs have been extended to defining dependencies that occur in a UML communication diagram [7].

The execution flow for the program and its PSDG is depicted in "Figure 4" below which shows the flow of control through the flamingo species class hierarchy and track's the species successors or predecessors according to the FDSA and BDSA algorithm respectively. The main() method is the starting execution run point for the programs. The classes are build in the hierarchical manner according to the inheritance relationships that they share with each other. For example, Class BirdsAves has only one successor- Phoenicopteriformes. The Phoenicopteriformes species has one successor as Phoenicopteridae and one predecessor i.e., BirdsAves. The Phoenicopteridae species has three successors as Phoenicopterus, Phoeniconaias and Phoenicoparrus, and two predecessors- one immediate and one indirect as Phoenicopteriformes and BirdsAves respectively. Moving further in the inheritance graph execution flow, the species Phoenicopterus has three successors as Great Flamingo, Carribean Flamingo and Chilean Flamingo with three predecessors- one immediate and two indirect as Phoenicopteridae, Phoenicopteriformes and BirdsAves repectively. Phoeniconaias has one successor as Lesser Flamingo and three predecessors- one immediate and two indirect as Phoenicopteridae, Phoenicopteriformes and BirdsAves repectively. Phoenicoparrus has two successors as Andean Flamingo and Jame's Flamingo and three predecessors- one immediate and two indirect as Phoenicopteridae, Phoenicopteriformes and BirdsAves respectiely. The Great, Carribean and Chilean Flamingo's have no successors but have common four predecessors- one immediate and three indirect as Phoenicopterus, Phoenicopteridae, Phoenicopteriformes and BirdsAves respectively. Similarly the Lesser, Andean and Jame's Flamingo's have no successor but Lesser Flamingo has four predecessors- one immediate and three indirect as Phoeniconaias, Phoenicopteridae, Phoenicopteriformes and BirdsAves respectively, and Andean and Jame's flamingo have four predecessors- one immediate and three indirect as Phoenicoparrus, Phoenicopteridae, Phoenicopteriformes and BirdsAves respectively.

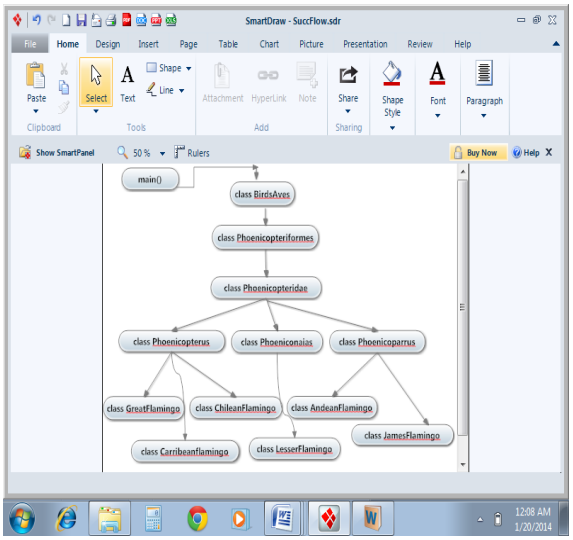
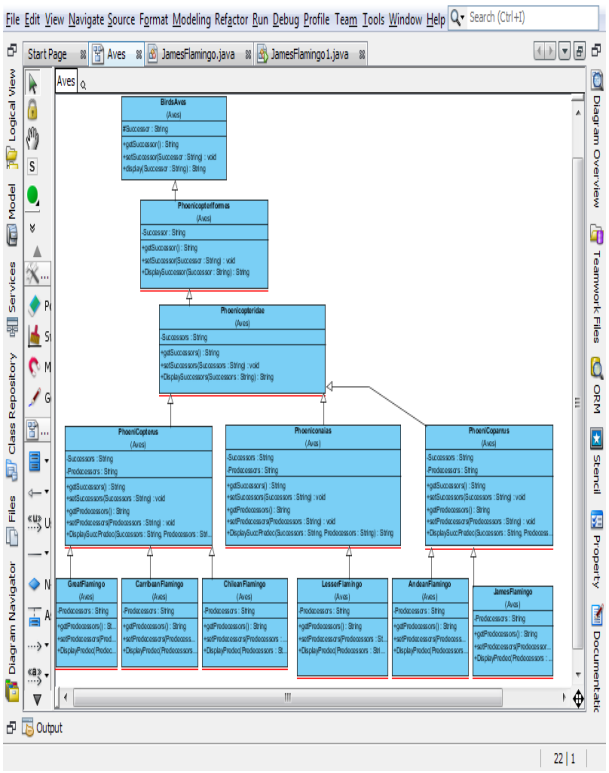
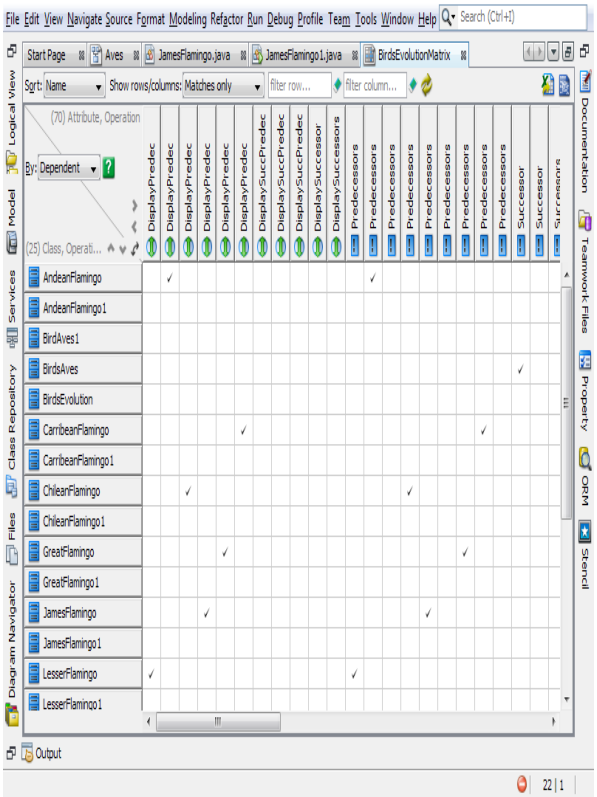


Fig. 4. Inheritance execution flow of evolutionary flamingo species

5. PROTOTYPE EXECUTION UPSHOTS

The seamless integration of UML into NetBeans ensures the maximization of the quality of source code and software system. It also minimizes the execution time and efforts involved in coding. The parallel compounding of the UML diagram with java source code enables automatic generation of source code from UML modeling diagram and vice versa. This makes software programming complacent with the modeling of the software. As the Software Development Lifecycle (SDLC) model requires coding of the software in accordance with the software model build, there remains gaps in the phases implementation of the SDLC. The solution to the above problem is the integration of the UML model construction with the programming IDE's (Integrated Development Environment). For the prototype execution of our FDSA and BDSA algorithm the execution has been performed in NetBean's IDE and the upshots are represented in "Figure 4" below. The first screenshot depicts the automatic generation of the UML class diagram from the source code and vice versa. The second and third screenshots represent the dependency matrix diagram showing the various dependencies existing between the flamingo birds species. This diagram also depicts the class hierarchy which is important for the inheritance track record.





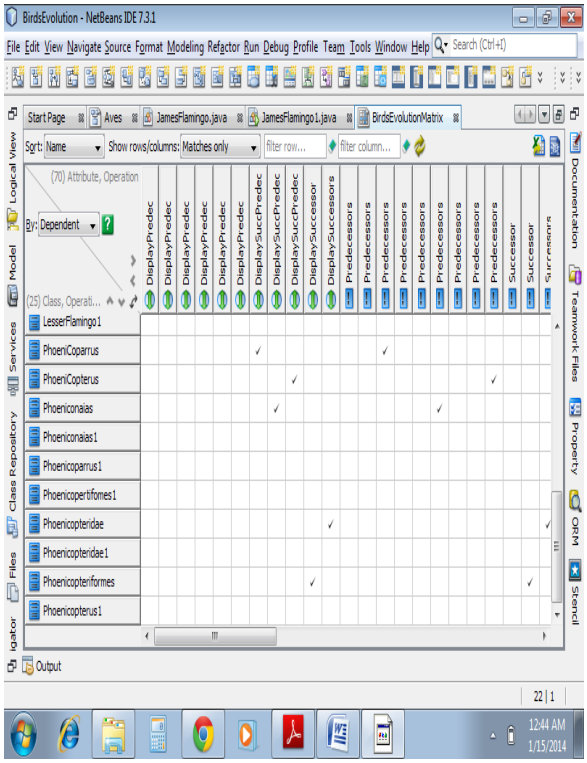


Fig. 5. Automatic generation of UML Diagram from Java Source Code and Dependency matrix between various classes in NetBeans IDE

6. EXECUTION RUN FOR FORWARD AND BACKWARD DYNAMIC SLIICING

The execution trace is implemented for both forward and backward dependencies. The LOC (FDSA) is calculated using the FDSA algorithm and the PSDG of "Figure 2". The LOC (BDSA) is calculated using the BDSA and the PSDG of "Figure 3". Both the dynamic execution trace results in precise slice with the reduction in statements of source code. Dynamic slicing has vast range of applications from debugging, cohesion measurement, testing and re-engineering [8]. These are an added advantage for the program comprehensions, debugging, and maintenance & reusability purposes.

The execution runs for the PSDG's of "Figure 2" and "Figure 3" is shown in "Table 1" below. The first column is for the species name tracing code which assigns the species with the tracing codes to be used in the slicing criterion.

The slicing criterion used is same for both the FDSA and BDSA algorithm as - dynamic slicing criterion<N, v, f, e>, where N= Statement number in the program, v= the affected variable, f=flamingo species, e= dynamic execution trace and are shown in column 3 and column 5 respectively.. With this dynamic slicing criterion the LOC's affected before and after FDSA and BDSA algorithmic implementation are shown in column 2, column 4 and column 6 respectively. The implementation results for LOC's that are calculated by applying FDSA and BDSA are almost half of the LOC's before the algorithm's execution which shows tremendous reduction in the program code. This would help in better program maintenance, debugging, computability and re-engineering. The dash in the rows represents those species that do not have ether a successor or predecessor. Thus, the FDSA and BDSA dynamic slicing algorithm helps in better traceability of the evolutionary species in the inheritance hierarchy.

Table 1. Execution Run Traces of FDSA and BDSA Algorithms in Java

Species name tracing code	**No of dependent lines of code (LOC)**	**Forward dynamic slicing criterion for successors**	**LOC (FDSA)**	**Backward dynamic slicing criterion for predecessors**	**LOC (FDSA)**	**Dynamic Precision attained**
BirdsAves (F1)	19	<11,C1, F1, Av>	8	-	-	*Yes*
Phoenicopteriformes (F2)	18	<25,C2,F2,S1>	8	<25,C2,F2,S1>	8	*Yes*
Phoenicopteridae (F3)	18	<39,C3,F3,S2>	8	<39,C3,F3,S2>	8	*Yes*
Phoenicopterus (F4)	18	<53, C4,F4,S3>	8	<53, C4,F4,S3>	8	*Yes*
Phoeniconaias (F5)	18	<67,C5,F5,S4>	8	<67,C5,F5,S4>	8	*Yes*
Phoenicoparrus (F6)	18	<81,C6,F6,S5>	8	<81,C6,F6,S5>	8	*Yes*
Greatflamingo (F7)	18	-	-	<95,C7,F7,S6>	8	*Yes*

Carribeanflamingo (F8)	18	-	-	<109C8,F8,S7>	8	*Yes*
Chileanflamingo (F9)	18	-	-	<123,C9,F9,S8>	8	*Yes*
Lesserflamingo (F10)	18	-	-	< 137,C10,F10,S9>	8	*Yes*
Andeanflamingo (F11)	18	-	-	<151,C11,F11,S10>	8	*Yes*
Jamesflamingo (F12)	18	-	-	< 165,C12,F12,S11>	8	*Yes*

7. CONCLUSION

The forward and backward dynamic slicing algorithm ascertains the dependencies between evolutionary organisms. This aids in figuring out the successor and predecessor relationship state existing by nature. The automated UML models are build from source code and vice-versa for validating the application development. The forward dynamic slicing algorithm traces the successors and backward dynamic slicing algorithm tracks the predecessors. Both algorithms are grounded on a dynamic slicing criterion which determines the dependencies between the hierarchical inheritances. The immediate & indirect successors and predecessors can be calculated with the execution run of the algorithm. Thus, the algorithm succeeds in ramifying phyletic flamingo species by retracing the phylogenetic ground. The efficiency of the algorithm is examined with varying inputs. This results in computation of precise dynamic slices in reduced LOC's and execution time. The algorithm is substantial for evolutionary species having complex phylogenetic ground. Thus, the taxonomic categorization can be channeled by incorporating the dynamic slicing algorithms at all the inheritance levels.

8. REFERENCES

[1] Donglin Liang and Mary Jean Harrold , *"Slicing Objects using System Dependence Graphs,"*International Conference on Software Maintenance, Washington, D.C, pp.358-67, November 1998.

[2] F. Tip. A Survey of Program Slicing Techniques, J ProgramrningL3nguages, 1995, 3(3):121-189.

[3] Baowen Xu Zhenqiang Chen, Dynamic Slicing Object-Oriented Programs for Debugging, Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'02), pp.115-122, 2002.

[4] Soubhagya Sankar Barpanda, Baikuntha Narayan Biswal, Sasmita Rani Behera, Mitrabinda Ray, Durga Prasad Mohapatra, Interprocedural Slicing of Generic Programs, IEEE International Conference on Signal Processing Systems, pp. 570-573, 2009.

[5] Zoltán Ujhelyi, Ákos Horváth, Dániel Varró, Towards Dynamic Backward Slicing of Model Transformations, IEEE ASE, pp. 404-407, Lawrence, KS, USALawrence, KS, USA, 2011.

[6] Jaiprakash T. Lallchandani R. Mall, Static Slicing of UML Architectural Models, Journal of Object Technology, Vol.8, No. 1, January-February, pp.159-188, 2009

[7] Alina Mishra, Subhrakanta Panda, Dishant Munjal, Dynamic Slicing of Aspect-Oriented UML Communication Diagram, International Journal of Computer Science and Informatics, Volume- 3, Issue- 2, pp. 58-63, 2013.

[8] N.Sasirekha, A.Edwin Robert and Dr.M.Hemalatha, Program Slicing Techniques and its Applications, International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, pp.50-64, July 2011.