Evaluate Imagix-4D Reverse Engineering Tool Capabilities

Ketki Balde[a], Dr. Lalji Prasad[b], and Dr. Rashmi Yadav[c]
a. Research Scholar, SAGE University, Indore, India, ketkibalde06@gmail.com
b. Professor & Head, SAGE University, Indore, India, hoi.iac@sageuniversity.in
c. Professor, SKITM, Indore, India, Rashmi.yadaver@gmail.com

Abstract
It is difficult to maintain software as its size and as well as complexity growing day by day, so, it is necessary to comprehend the many components of a complex software system. For understanding software systems or legacy code, we need reversing engineering tools. Various reverse-engineering tools are available each one has a different working style. The majority of tools only show the static, not the dynamic, behavior of the software system. We found that Imagix 4d is used to visualize the static behavior and some extent to dynamic aspects of the software system. In the paper, we describe Imagix 4d's features, overview,merits, and demerits, and also analyze the architecture generated by it.
Keywords: Reverse Engineering, Imagix 4D, static behavior, dynamic behavior, visualization e.t.c.
DOINumber:10.48047/nq.2022.20.22.NQ10273 **NeuroQuantology2022;20(22):2802-2816**

Introduction
We use the latest tools and techniques to develop the new project, but with time these tools and techniques become old and new ones come to the market. We can easily replace the old hardware with the new one, but in the case of software, it is very difficult to replace the old one, so we have to build the new software or enhance it, but enhancing the software is also very difficult task if we don't its code. Here the reverse engineering comes.[1]
It basically analyzes the code and recovered the following information like design documents, requirement specifications, and functions of a product. It builds adatabase of code and recovers essential information from this. Primarily it is used to understand the legacy code. By making a system easier to understand and producing the data needed for a legacy system, reverse engineering's primary goal is to accelerate maintenance work. [2]
Three steps comprise reverse engineering. Extraction of information, modelling, and review. In information extraction,the software information about its design is inspected and extracted. This information is then utilized to locate the software's relevant documents. The information collected in the first step is used to design models in the modeling step. The models may be a class diagram, data flow diagram, etc. depending upon the tool we are going to use. In the last step, we review the model designed in the modeling step with already available information and checked whether we get the desired output or not. [2]
The application areas for reverse engineering and reengineering tools include software maintenance, re-documentation, design recovery, metric analysis, restructuring, source code translation, migration to another hardware platform/operating system, objectification, and reusable component extraction [1]. Our work has a positive social impact because it makes tasks easier to complete for developers and maintainers while also saving them money and time.
For reverse engineering of software, various tools are available based on different methods are rigi, imagix 4d, solidfx, solidsx, columbs, e.t.c.. Each tool has its unique working style

different from one another. These tools analyze the source code using static analysis to extract information, which involves a number of activities such as fact extraction from the code, fact aggregation, and an interface to show the information.[3].

Comprehending the historical code sequence diagram is also very critical.Among the various available tools, Imagix 4d has a very good user interface and it also visualizes the software in the form of flow charts, flow graphs, class diagrams, and, to a lesser extent, dynamic visualizations.[1].

In this paper,we describe about the imagix 4d tool, its working, features, merits, and demerits. It is used to analyze large C, C++, and java code. We select imagix 4d because it generates maximum design from source code and in spite of this it also has various features. This tool helps developers to understand the code and also helps in improving the complex third-party code.Various tasks, including the study of dependencies and control flows, are also automated using it. It is also able to identify the problems in concurrency and data sharing. By using imagix 4d productivity and quality of any software system increase. It also helps in reducingthe risk. Another main feature of this code is that it simultaneously displays code and a visual window.[2].

We addressed related work for imagix 4d in section 2, provided an overview of imagix 4d in section 3, and then examined the architecture that imagix 4d produced in section 4.

2 Related Work:

Various authorshave done remarkable studies for imagix 4D. The description of their studies is as follows:

In [1] authors, said that imagix 4d generates the class diagram, flow chart, and graphs for c, c++, and java software. They took some small codes and analyze imagix 4d and observed that it generates a flow chart in three ways. In a first way, it generates a flow chart but does not display the coding details. In a second way, it displays code with a flow chart. In a third way, it displays the line number with code in the flow chart. It also displays the separate flow chart for different functions in a single code. The generated flow chart is very flexible in nature.[1]

The most important feature of imagix 4d is matrix visualization. It is very much needed for analyzing the design of code, for reverse engineering, and for testing. They also said that there is no error detection and correction in imagix 4d. It does not display the flow chart for extended and advanced loops like the if else statement. Additionally, it creates a class diagram that is abstract in nature and does not show the variables' data types and the access modifiers for the functions. This knowledge is crucial for reverse engineering. [1]

In [4], authors said that imagix 4d is used for large and complex software and also used for unfamiliar or legacy code. The diagram generated by imaix 4d is in more readable form, and it generates this information ina 3D graphical format which is very helpful for the user. It also has a very good parser which is useful for reverse engineering. The primary benefits of imagix 4d is that it automatically generates the documentation from the source code. It also has a very good UI, by which users can do a proper study of software ranging from high-level abstraction to low-level abstraction like classes, and functions.[4]

It also generates charts and graphs of code. Graphs also have zooming functions that help user to visualize graphs from all directions. Graphs also have grouping functions that allow users to group similar functions. By this grouping, graphs become more readable and understandable. It provides a very huge database that can store huge software. The tool works with several operating systems, including HP-UX, Linux, Solaris, and Windows XP/Vista/Win7/Win8.Generated graphs are very small, so closely connected graphs get complicated and unreadable. It does not provide a layer view in the graph and also the user is not able to save the grouping in a graph.[4]

In [5] According to the author, a significant imagix 4d shortcoming is that it does not provide the sequence diagram of a code, which is also crucial when doing reverse engineering for code improvement. A sequence diagram is highly helpful since it shows how items interact throughout a specific time frame. It does not generate the ER diagram and state machine diagram, they both are very important for the analysis of code. [5]

In [6], authors said that imagix 4d has versatile reverse engineering functionality, skilled visualizations, and selection feature. With class diagram and call graph feature, it also generates Intra-procedural data flow diagrams and also generates documentation from source code. It is able to generate a 3D view of graphs. Generated graphs have 2-dimensional and 3-dimensional features. It uses UML notations for the class diagrams. It is able to generate a data flow diagram that contains if else statements, functions call, or based on the user's choice. The main feature of the call graph generated is that it hasa link option that we can go from the call graph element to corresponding elements in the source code.[6]

In [7], the authors said that imagix 4d is used for parsing of C++ templates, but the important information of c++ code has not been extracted by imagix parser like information related to function calls. It also correctly identifies constructor calls but is not able to identify the dependencies among the type of argument and virtual functions. It is also able to browse, analyze, and navigate the c/c++ source code. Only a template argument can be used by Imagix-4D to denote an invocation; however, the function to which the invocation resolves cannot be resolved at compilation time.[7]
On the basis of the studies of various authors, we find out some common features of imagix, which are shown in fig. 1.

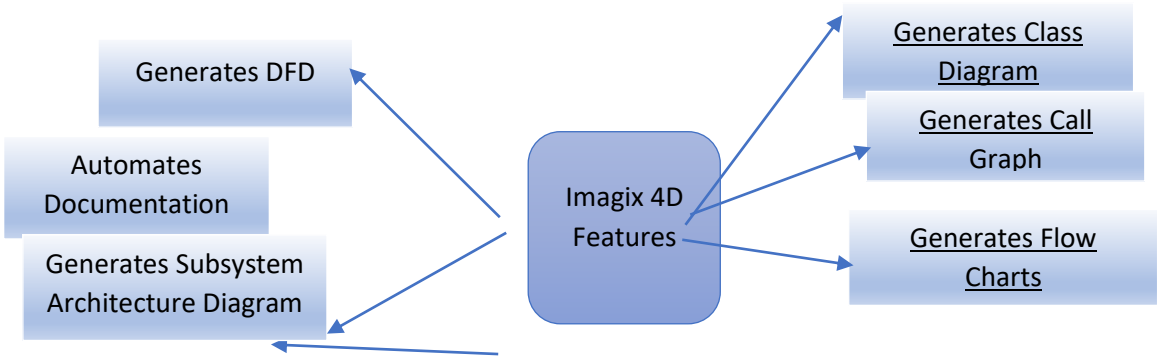


Fig.1 Features of Imagix 4D

3 Overview:
Imagix 4D helps the user to boost their development, testing, reuse, and maintenance of software. It also helps users to learn unfamiliar code and help to solve errors. It also displays the high-level architecture of a system that includes components, dependencies among components, and packaging. It uses abstract form to represent high-level architecture for example in code if function x of class x calling function y of class y, then it summarized it in an abstract way like class x calling class y. High-level architecture includes Subsystem Architecture Diagrams, Design Structure Matrices (DSM), UML Class Diagrams, Class Inheritance view, Files Include Hierarchy view, and UML Task Collaboration Diagrams. [1]
Subsystem Architecture Diagrams provided by imagix 4D help in visualization and analyzing the architecture of the software. To understand the software and its design this diagram is very much needed. It also provides a delta analysis of the code. Delta analysis is the analysis between the current version of the software with the previous version at a structure level. By delta analysis, we can also compare the current structure of the project with the current structure of another project that we had created before. [3]
The main feature of imagix 4D is to generate a class diagram. Imagix class diagram includes classes, interfaces, and their relationships i.e., inheritance, aggregation, and association. The class diagram clearly resembles the inheritance like which classes inherit from the other classes. The class diagram also shows the data member and member function clearly. There is another feature of imagix class diagram is to display associations on or off and we can also select association type. By the graphical querying feature of the graph, users can customize the class diagram. Users can focus on a particular

section of the class diagram which he wants to examine. Users can also control the number of classes in a class diagram and also control which classes appear in the class diagram and which are not. Users can also control which classes are expanded and which are reduced. [5]

Imagix 4D also generates a task collaboration diagram, which provides information about the shared variable. For multitasking and multithreaded system, the information regarding a shared variable is very important. In a multitasking system, the user has to define tasks, but in c,c++ and java don't allow the user to do so. This facility is provided by the imagix 4D task definition dialogue feature. Using this feature user can define tasks. [7]

Imagix 4D analyze source code and first provides information about the high-level structure then the mid-level and thenthe low-level structure of code.The mid-levelarchitecture includes information about function calls, variables, information about macro use by function, classes used, and also the data type hierarchy. The low-level architecture includes information about the sequence diagram and flow chart. [8]

4 Architecture analysis created with Imagix 4D:

As we discussed in the previous section that imagix 4D generates various diagrams from source code such as flow charts, flow graphs, and class diagrams, in this section we analyzed all these diagrams generated by imagix 4D. For this, we took small and as well as large codes of C,C++, and java programming languages.

Class diagram, flow chart, flow graph all are necessary diagrams for forward as well as for reverse engineering. The class diagram is a static diagram that depicts the static perspective of an application. In the class diagram, a number of classes, interfaces, associations, collaborations, and constraints are displayed. Additionally, it explains a class's characteristics and functions. For the purpose of examining the class diagram created by imagix 4D, we took some c/c++ and java scripts.

The generated diagrams for the program (a) are displayed in Fig. 2.1.and in Fig. 2.2. The imagix class diagram has two variants. One shows the number of attributes and operations, but doesn't display its name. and another one display the name of operations and variables. But it also does not display the data type of variables and return type of the method. These are all the internal information of any code and this is very important when we do reverse engineering. So, it must be necessary to display the internal information of any code by a class diagram.

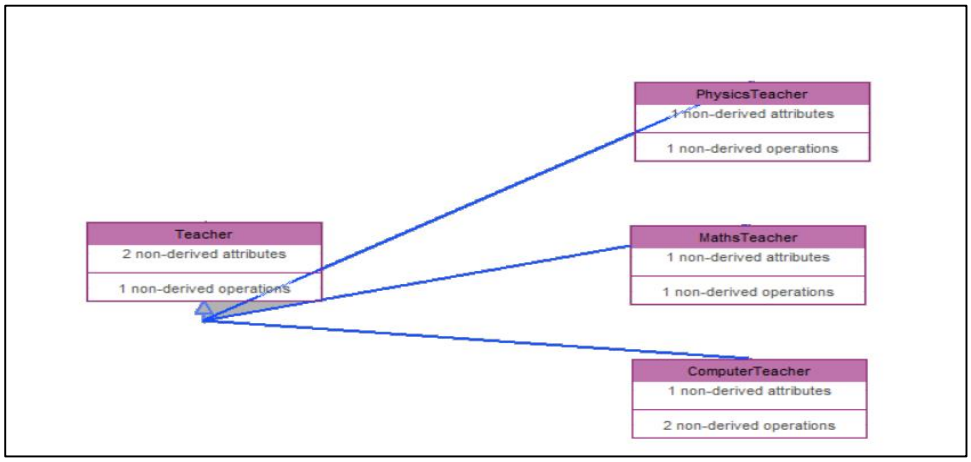


Fig 2.1. Class Diagram created by imagix 4D. (Variant 1)

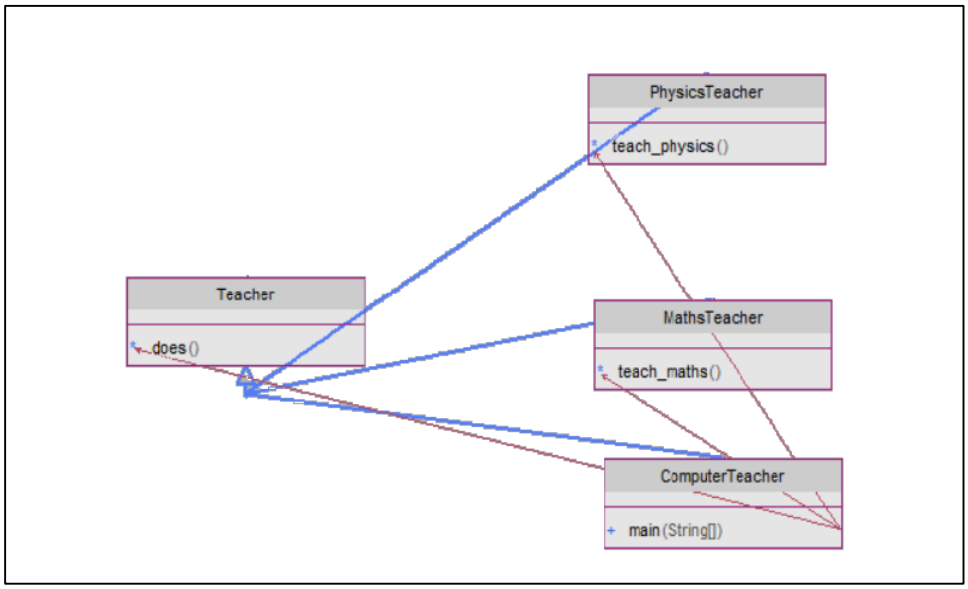


Fig 2.2. Class Diagram created by imagix 4D. (Variant 2)

Other than class diagrams Imagix also generates some basic diagrams from the code like class inheritance diagrams, class call diagrams, datatype hierarchy diagrams. Fig. 3 shows the class inheritance diagram. This is a simple architecture of inheritance used in code. Through this diagram, the user gets information related to the parent class and its child classes. In this diagram teacher is a parent class and physics teacher, maths teacher and computer teacher are child classes. Fig.4 shows the class call diagram. Through this diagram, users know about the calling relationship of a code. In the diagram, the computer teacher calls the method of a maths teacher, physics teacher, and teacher.

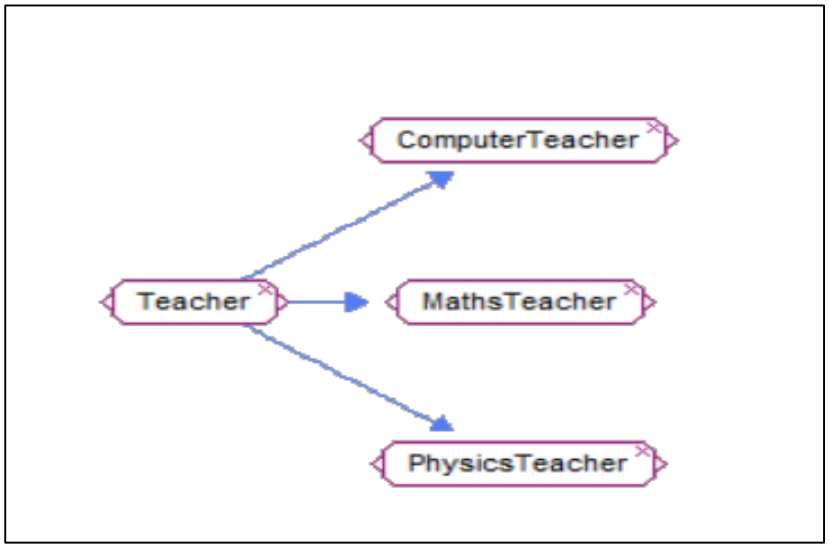


Fig 3. Class Inheritance Diagram created by imagix 4D

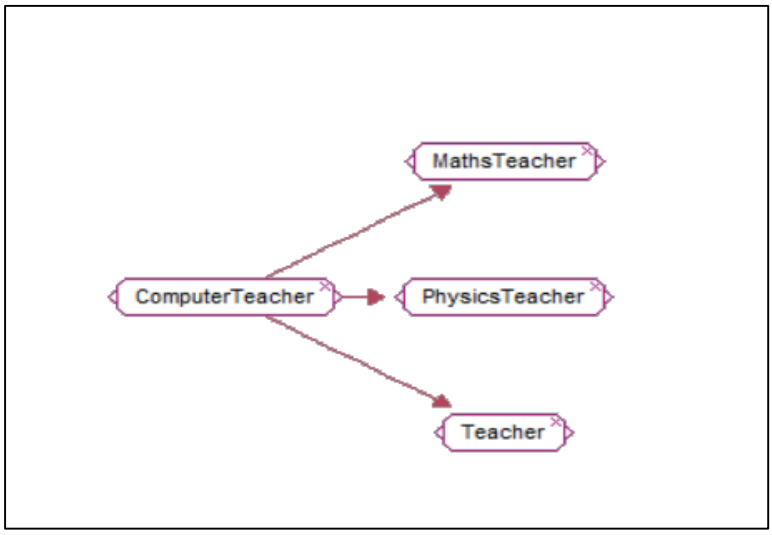


Fig 4. Class Call Diagram created by imagix 4D

Imagix class diagram also displays the association among classes. It displaysthe type of relationship between classes like one to one, one to many, and also displays aggregation and composition among classes. Fig 5 shows the association among classes in the class diagram for the given program (a). Fig 6 shows the datatype hierarchy diagram, this diagram gives information regarding the hierarchy of classes in which they are created.

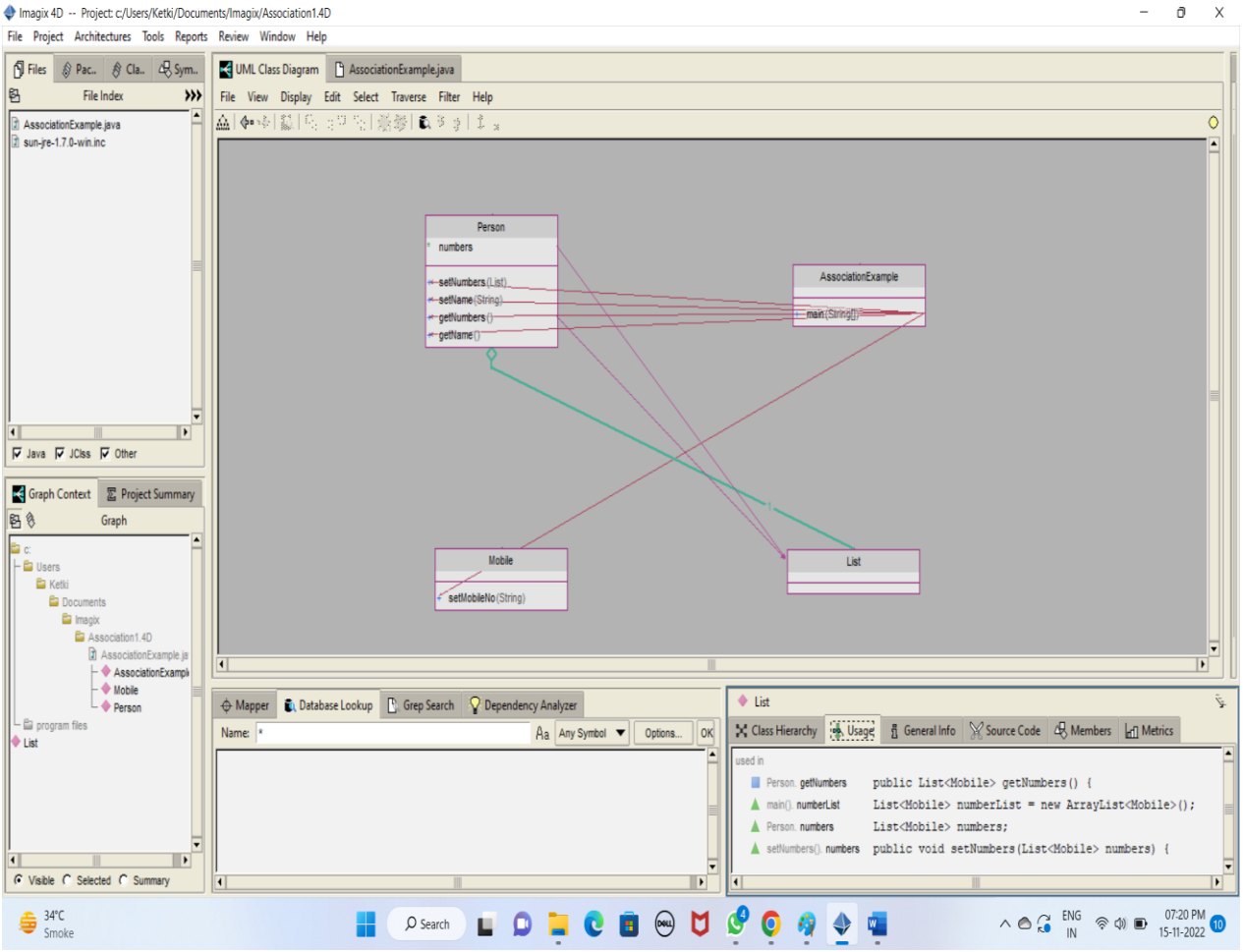


Fig 5. Class Diagram generated by imagix 4D (aggregation)

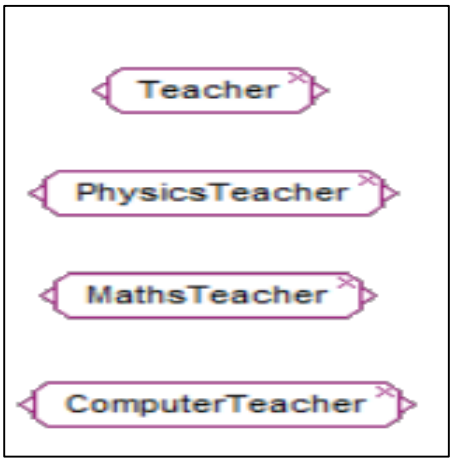


Fig 6. Datatype Hierarchy Diagram generated by imagix 4D

Other than class diagram it also generates diagram related to the function or methods of any code. Fig 7. Shows the function call diagram for the given program (a). This diagram gives information related to the function call and its classes. By this diagram, we easily know about the class in which we define the main method and also the functions of classes. This diagram is also very essential to know about the all methods of a class when we do reverse engineering.

Fig 8. shows the function call with a variables diagram. This diagram gives detailed information regarding the methods and variables of respective classes. Square represents the methods of a class and trapezoid represents the variable and objects of a class. When we do reverse engineering this diagram gives the overview of the objects, variables, and methods of a class, this will help in the understanding of a code.

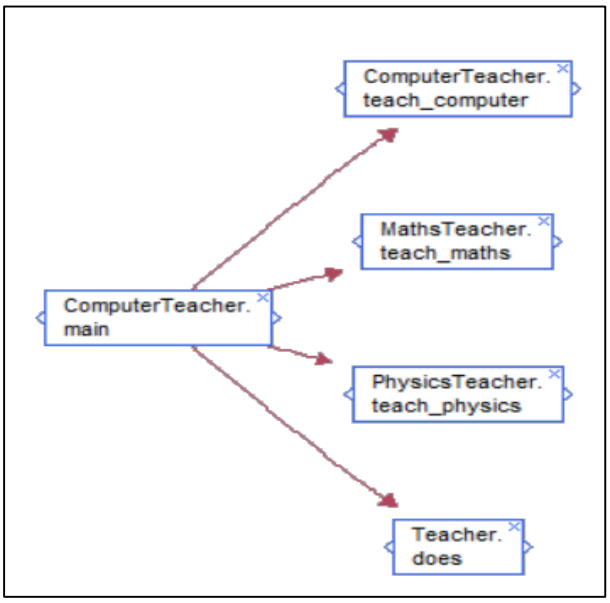


Fig 7. Function Call Diagram generated by imagix 4D

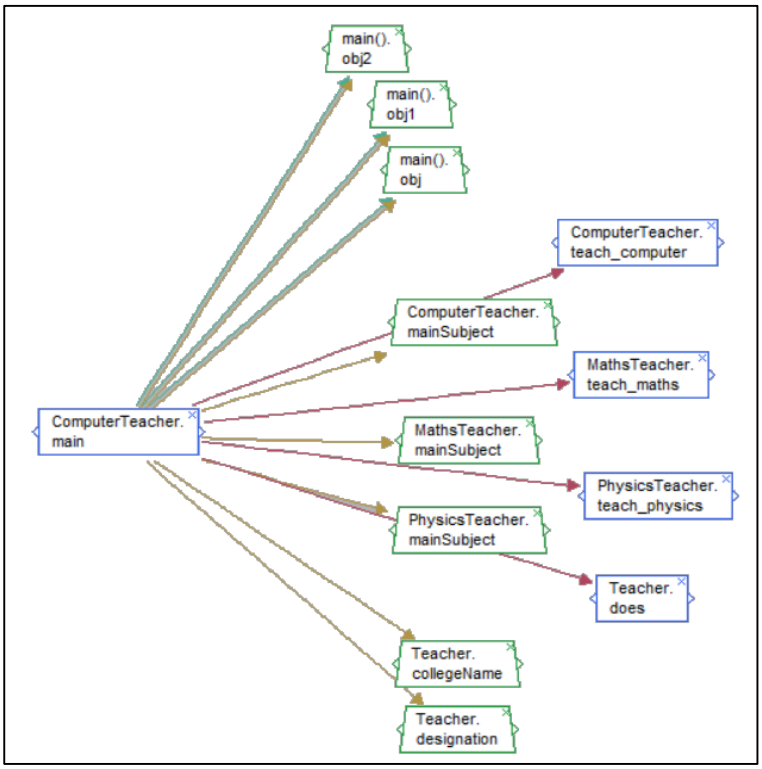


Fig 8. Function Call with variables Diagram generated by imagix 4D

One of the main feature of imagix is to generate a flow chart for any application. The flow chart is a pictorial representation of the steps performed and decisions taken for the execution of any task in the application. In the flow chart, a different shape is used for each step. The diamond shape is used for any decision, the square shape is used for process/code, and the parallelogram shape is used for input/output. It gives a detailed description of a code from beginning to end. it communicates the steps in a process very effectively and efficiently. The flow chart generated by imagix for the given program (b) is shown in fig 9.

Imagix creates flowcharts in three different formats: without source code, with source code, and with line numbers. For the large code flow chart with source code become complex and big, so for large lines of code flow chart without source code is used, and when user want to understand the detailed logic with conditions then flow chart with source code is used.

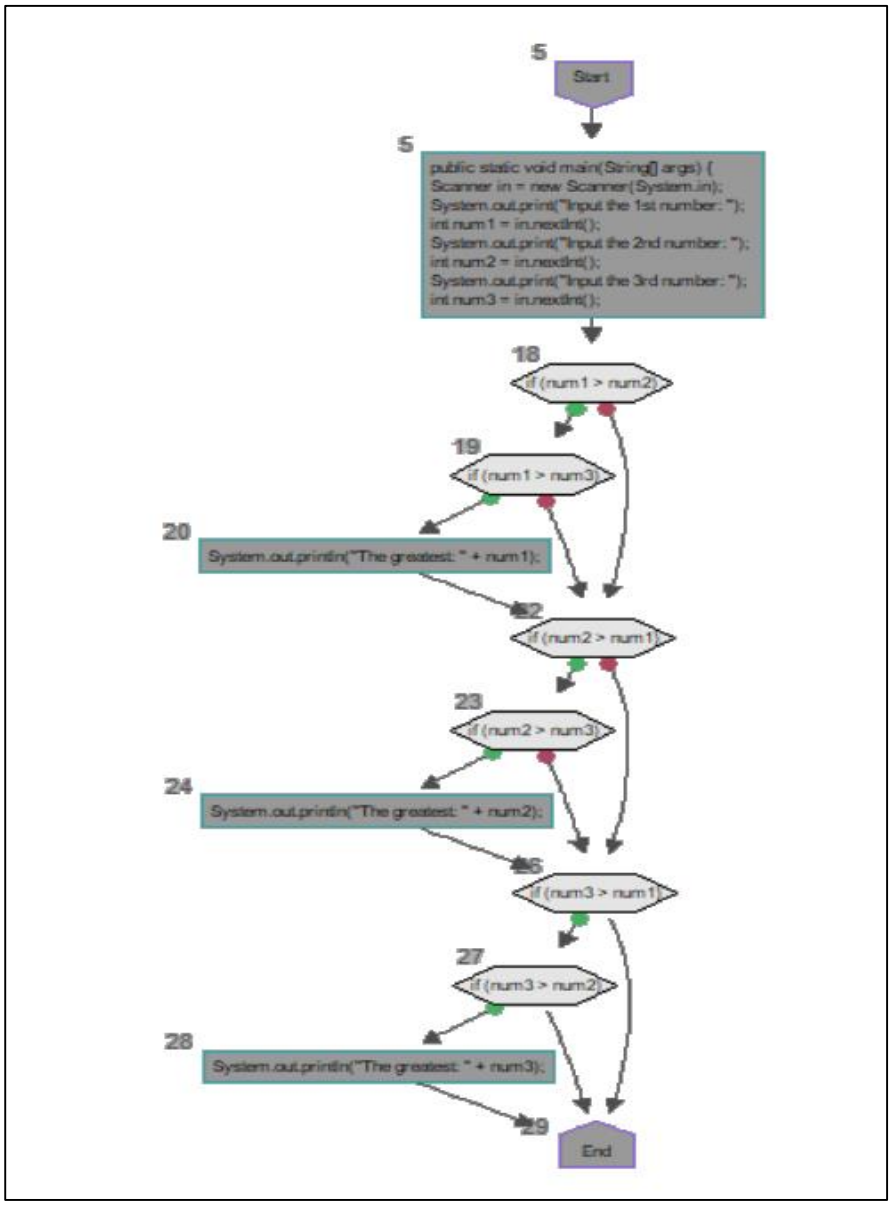


Fig 9. Flow chart generated by imagix 4D with source code and line number.

Imagix also generates the sequence diagram. Fig 10 shows the sequence diagram for the program (c). A sequence diagram depicts the overall structure of a program in a particular sequence. It gives the dynamic visualization of any application. It depicts the interaction between different application users over time. It is used to investigate the high-level interactions between various objects in any system. It demonstrates how operations are carried out, what messages are sent, and when they are sent. Imagix's generated sequence diagram differs from the actual sequence diagram defined by the unified modeling language.

On the basis of study we have observed that it have some limitation It shows a sequence diagram that look like a class diagram. Although, it depicts how various application actors interact with one another but doesn't display their time frame.

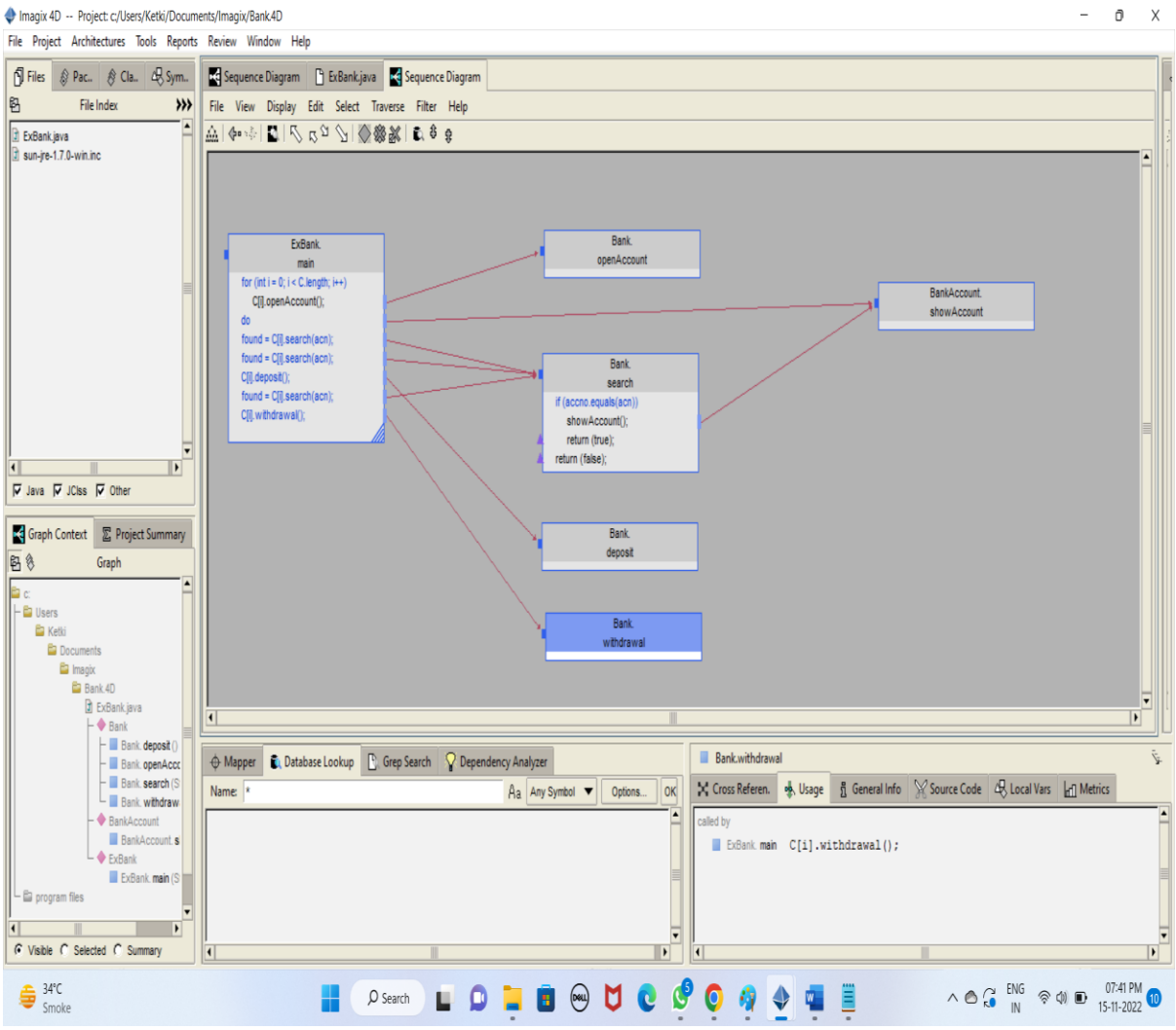


Fig 10. Sequence Diagram generated by imagix 4D.

5 Discussion& Conclusion:

Imagix 4D is a good choice among the different tools available since it generates a maximum diagram and offers filtering choices so that users may view implementation in accordance with their preferences. It also supports multiple languages, including C, C++, and Java. Reverse engineering is the process of disassembling an object to discover how it functions. It is used to see the working of any objects, generally, it is used to rebuild or enhance an object. The majority of tools are designed to create static information about software, such as class diagrams and state diagrams, rather than dynamic information, such as sequence diagrams that depict how objects interact.

The class diagram produced by imagix shows the most relevant data about a code that is necessary for reverse engineering, but it omits internal code specifics. The tool needs to be enhanced in this area. It doesn't verify the code for errors if there are any, it nevertheless generates the class diagram. Consequently, the parser also needs to be enhanced.Imagix only shows static code information, not dynamic data resulting from user interaction. The dynamic information of a code is displayed in sequence diagrams and ER diagrams. Although it partially supports the sequence diagram but does not provide the ER diagram. Any application's dynamic information is essential for code improvement and reverse engineering. In order to define the requirement, set for a fully-functioning, all-inclusive reengineering/reverse engineering tool, this study has concentrated research efforts on producing a completely functional reverse engineering tool.

Appendix (Program Used):

a) Inheritance Program in Java

```
class Teacher {
 String designation = "Teacher";
 String collegeName = "NIT";
 void does(){ System.out.println("Teaching"); }}
 class PhysicsTeacher extends Teacher{
 String mainSubject = "Physics";
 void teach_physics()
{ System.out.println("Teaching Physics");}}
class MathsTeacher extends Teacher{
String mainSubject = "Maths";
void teach_maths()
{ System.out.println("Teaching Maths");}}
class ComputerTeacher extends Teacher{
String mainSubject = "Computer";
void teach_computer()
{ System.out.println("Teaching Computer");}
public static void main(String args[]){
PhysicsTeacher obj = new PhysicsTeacher();
System.out.println(obj.collegeName);
System.out.println(obj.designation);
System.out.println(obj.mainSubject);
obj.does();
obj.teach_physics();
MathsTeacher obj1 = new MathsTeacher();
System.out.println(obj1.collegeName);
System.out.println(obj1.designation);
System.out.println(obj1.mainSubject);
obj1.does();
obj1.teach_maths();
ComputerTeacher obj2 = new ComputerTeacher();
System.out.println(obj2.collegeName);
System.out.println(obj2.designation);
System.out.println(obj2.mainSubject);
obj2.does();
obj2.teach_computer(); }}
```


b) Greater number among three numbers in java

```
import java.util.Scanner;
public class GreaterNum {
public static void main(String[] args) {
Scanner in = new Scanner(System.in);
System.out.print("Input the 1st number: ");
int num1 = in.nextInt();
System.out.print("Input the 2nd number: ");
int num2 = in.nextInt();
System.out.print("Input the 3rd number: ");
int num3 = in.nextInt();
```

```
if (num1 > num2)
if (num1 > num3)
System.out.println("The greatest: " + num1);
if (num2 > num1)
if (num2 > num3)
System.out.println("The greatest: " + num2);
if (num3 > num1)
if (num3 > num2)
System.out.println("The greatest: " + num3);}}
```

c) Program of bank operations in java

```
import java.util.Scanner;
public class ExBank  {
public static void main(String arg[]) {
Scanner KB = new Scanner(System.in);
System.out.print("How Many Customer U Want to Input : ");
int n = KB.nextInt();
Bank C[] = new Bank[n];
for (int i = 0; i < C.length; i++) {
C[i] = new Bank();
C[i].openAccount();}
int ch;
do {
System.out.println("Main Menu\n1. Display All\n 2. Search By Account\n 3. Deposit\n 4. Withdrawal\n 5.Exit ");
System.out.println("Ur Choice :");
ch = KB.nextInt();
switch (ch) {
case 1:
for (int i = 0; i < C.length; i++) {
C[i].showAccount();}
break;
case 2:System.out.print("Enter Account No U Want to Search...: ");
String acn = KB.next();
boolean found = false;
for (int i = 0; i < C.length; i++) {
found = C[i].search(acn);
if (found) {
break;}}
if (!found) {
System.out.println("Search Failed..Account Not Exist..");
break;
case 3:
System.out.print("Enter Account No : ");
acn = KB.next();
found = false;
```

```
for (int i = 0; i < C.length; i++)

{found = C[i].search(acn);
if (found) {
C[i].deposit();
break; }
if (!found) {
System.out.println("Search Failed..Account Not Exist.."); }
break;
case 4:System.out.print("Enter Account No : ");
acn = KB.next();
found = false;
for (int i = 0; i < C.length; i++){
found = C[i].search(acn);
if (found){
C[i].withdrawal();
break;} }
if (!found){
System.out.println("Search Failed..Account Not Exist..");}
break;
case 5:System.out.println("Good Bye..");
break;}}
while (ch != 5);}}
class BankAccount{
public String accno;
public String name;
public long balance;
void showAccount(){
System.out.println(accno + "," + name + "," + balance);}}
class Bank extends BankAccount{
Scanner KB = new Scanner(System.in);
void openAccount(){
System.out.print("Enter Account No: ");
accno = KB.next();
System.out.print("Enter Name: ");
name = KB.next();
System.out.print("Enter Balance: ");
balance = KB.nextLong();}
void deposit(){
long amt;
System.out.println("Enter Amount U Want to Deposit : ");
amt = KB.nextLong();
balance = balance + amt;}
void withdrawal(){
long amt;
System.out.println("Enter Amount U Want to withdraw :
");
amt = KB.nextLong();
if (balance >= amt){
balance = balance - amt;}
else {System.out.println("Less Balance..Transaction
```

```
Failed..");}}
boolean search(String acn){
if (accno.equals(acn)){
showAccount();
return (true);}
return (false);}}
```

References:

[1]. Rashmi Yadav, Ravindra Patel and Abhay Kothari," Critical evaluation of reverse engineering tool Imagix 4D!" SpringerPlus (2016) 5:2111.

[2]. Roger S. Pressman, 2010," Software Engineering A Practitioner's Apprach" Seventh Edition, McGraw-Hill.

[3]. Alexandru Telea, Heorhiy Byelas, "A Framework for Reverse Engineering Large C++ Code Bases", Electronic Notes in Theoretical Computer Science 233 (2009) Elsevier.

[4]. Jayanti Upadhyay, Lalji Prasad," Study of Reverse Engineering and Assessment of RIGI and Imagix 4D" International Journal of Computer Application (2250-1797) Volume 5– No. 5, August 2015.

[5]. Rashmi Yadav, Ravindra Patel and Abhay Kothari, "Reverse Engineering Tool Based on Unified Mapping Method (RETUM): Class Diagram Visualizations" Journal of Computer and Communications, 2014.

[6]. Jussi Koskinen1, Tero Lehmonen, "Analysis of Ten Reverse Engineering Tools ", Advanced Techniques in Computing Sciences and Software Engineering, Springer Science+Business Media B.V. 2010.

[7]. Thomas Gschwind, Martin Pinzger, Harald Gall, "TUAnalyzer—Analyzing Templates in C++ Code", Conference on Reverse Engineering (WCRE'04) 1095-1350/04, 2004 IEEE.

[8]. SolidSource BV, Eindhoven, "The Solid* toolset for software visual analytics of program structure and metrics comprehension: From research prototype to product", Science of Computer Programming 79 (2014) 224–240.

[9]. Rudolf Ferencl, Arpad Beszedes, Mikko Tarkiainen and Tibor Gyimothy, "Fact Extraction and Code Auditing with Columbus and SourceAudit", Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04).

[10]. Alexandru Telea, Lucian Voinea, "SOLIDFX: An Integrated Reverse Engineering Environment for C++", 12th European Conference on Software Maintenance and Reengineering, IEEE 978-1-4244-2157,2008.

[11]. Martin Habringer, Michael Moser and Josef Pichler, "Reverse Engineering PL/SQL Legacy Code: An Experience Report", International Conference on Software Maintenance and Evolution, IEEE 2014.

[12]. Holger M. Kienle, Hausi A. Muller," Rigi—An environment for software reverse engineering, exploration, visualization, and re-documentation" Science of Computer Programming 75 (2010) 247–263 elsevier.

[13]. D. Reniers, L. Voinea1 and A. Telea," SolidSX: A Visual Analysis Tool for Software Maintenance", Poster Abstracts at Eurographics/ IEEE-VGTC Symposium on Visualization (2010).

[14]. Walter R. Bischofberger, "Sniff—A Pragmatic Approach to a C++ Programming Environment", USENIX C++ conference, Portland, Oregon, August 1992.

[15]. Gunter Fleck, Wilhelm Kirchmayr, Michael Moser, Ludwig Nocke, and Josef Pichler," Experience Report on Building ASTM Based Tools for Multi-Language Reverse Engineering", IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering,2016.

[16]. Ahmed Saleem Abbas and DR. W. Jeberson, "Proposed Software Re-engineering Process That Combine Traditinal Software Reengineering Process With Spiral Model", International Journal of Advanced Research in Computer Science, 2013.

[17]. Dragan BOJIC, Dusan VELASEVIC, "A Classification Of Reverse Engineering Tools

And Criteria For Their Evaluation And Selection", Yugoslav Journal of Operations Research, 1999.

[18]. Derek Rayside, Marin Litoiu and Margaret-Anne Storey, "Visualizing Flow Diagrams in WebSphere Studio Using SHriMP Views", Information Systems Frontiers 5:2, 161–174, 2003.

[19]. M.A.D. Storeyyz K. Wongy P. Fongz D. Hooperz K. Hopkinsz H.A. Muller," On Designing an Experiment to Evaluate a Reverse Engineering Tool", Proceedings of WCRE '96: 4rd Working Conference on Reverse Engineering,IEEE.

[20]. H. J. S. Basten, P. Klint, "DeFacto: Language-Parametric Fact Extraction from Source Code", International Conference on Software Language Engineering,2008.

[21]. Michele Lanza, "CodeCrawler — Lessons Learned in Building a Software Visualization Tool", Seventh European Conference on Software Maintenance and Reengineering, 2003. Proceedings., IEEE 2003.

[22]. Volker Riediger, "Analyzing XFIG with GUPRO", Proceedings Seventh Working Conference on Reverse Engineering, IEEE 2002.

[23]. Matthias Merdes, Dirk Dorsch, "Experiences with the Development of a Reverse Engineering Tool for UML Sequence Diagrams: A Case Study in Modern Java Development",Proceedings of the 4th International Symposium on Principles and Practice of Programming in Java, PPPJ 2006.

[24]. Eleni Strouli, Tarja Systa, "Dynamic Analysis For Reverse Engineering and Program Understanding", ACM SIGAPP Apllied Computing Review, Volume 10, Issue 1, Spring 2002.

[25]. Rudolf Ferencl, Arpad Beszedes, Mikko Tarkiainen and Tibor Gyimothy, "ColumbusReverse Engineering Tool and Schema for C++", International Conference on Software Maintenance, IEEE 2002.

[26]. M.H. Aabidi, Bouziane El Mahi, Chafik Baidada, Abdeslam Jakimi, and Hany Ammar, "Benefits of reverse engineering technologies in software development makerspace ", ITM Web of Conferences 13, (2017).

[27]. Aman Jatain and Deepti Gaur, "Reverse Engineering of Object Oriented System using Hierarchical Clustering", International Arab Journal of Information Technology · September 2017.

[28]. Aoun Raza, Gunther Vogel, and Erhard Pl¨odereder, "Bauhaus – a Tool Suite for Program Analysis and Reverse Engineerin", International Conference on Reliable Software Technologies, Springer, 2006.

[29]. A. Lee, A. Payne, T. Atkison, "A Review of Popular Reverse Engineering Tools from a Novice Perspective", International Conference of Software Engineering Research and Practice , SERP'18.

[30]. H. A. Muller, J. H. Jahnke, D. B. Smith, M.-A. Storey, S. R. Tilley, ¨ and K. Wong, "Reverse engineering: A roadmap," in Proceedings of the Conference on the Future of Software Engineering. ACM, 2000, pp. 47–60.

[31]. G. CanforaHarman and M. Di Penta, "New frontiers of reverse engineering," in 2007 Future of Software Engineering. IEEE Computer Society, 2007, pp. 326–341.

[32]. M. Hafiz and M. Fang, "Game of detections: how are security vulnerabilities discovered in the wild?" Empirical Software Engineering, vol. 21, no. 5, pp. 1920–1959, 2016.

[33]. https://www.imagix.com/products/source-code-analysis.html

[34]. https://www.imagix.com/appnotes/uml-class-diagrams.html

[35]. https://www.imagix.com/products/source-code-analysis.html