

Reverse Engineering Tool Based on Unified Mapping Method (RETUM): Class Diagram Visualizations

Rashmi Yadav¹, Ravindra Patel², Abhay Kothari³

¹Department of Computer Science & Engineering, Acropolis Technical Campus, Indore, India

²Department of Computer Application, RGTU, Bhopal, India

³Department of Computer Science & Engineering, Acropolis Institute of Technology & Research, Indore, India
Email: rashmi.yadaver@gmail.com, ravindra@rgtu.net, kothari_abhay21@yahoo.com

Received 28 July 2014; revised 1 September 2014; accepted 2 October 2014

Copyright © 2014 by authors and Scientific Research Publishing Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

In this research paper, we evaluate an assortment of tools and intend to investigate multifarious characteristic of Imagix-4D Reverse Engineering Tool and on the basis of investigation find out inadequacy of Imagix-4D Reverse Engineering Tool (illustrate only abstract Class Diagram, and it has no support to illustrate ER-Diagram and Sequence Diagram) and propose a Reverse Engineering Tool based on Unified Mapping Method (RETUM) for prominence of Class Diagram Visualizations which surmount the limitation (class diagram which is intricate in visualization) of Imagix-4D Reverse Engineering Tool.

Keywords

Reverse Engineering, Imagix-4D Reverse Engineering Tool, Class Diagram

1. Introduction

Understanding the intricate relationships that exist between the source code components of a software system can be an arduous task. In the preceding years, several tools [1] have emerged to support program understanding, software maintenance, reverse engineering, and reverse engineering activities. A large part of such tools extract their information mainly from the source code via static analysis. This includes a set of operations ranging from code parsing and fact extraction, fact aggregation and querying, up to interactive visualization. Many requirements were met through previous Reverse Engineering Tool that was accepted by software industry, for designing purpose. Reverse Engineering Tool (**Figure 1**) is important for Lexical analyzer or scanner, its function to read the source program in the form of character stream and also grouping the logically related characters together

that are known as lexemes. Syntax analysis: parser uses the token_name taken from the token stream to generate the output in the form of a tree-like structure known as syntax tree or parse tree and semantic analysis: semantic analyzer uses the parse tree and symbol table for checking the semantic consistency of the language definition of the source program. The main function of the semantic analysis is type checking in which semantic analyzer checks whether the operator has the operands of matching type. Next phase is intermediate code generation phase: in intermediate code generation phase, the parse tree representation of the source code is converted into low-level or machine-like intermediate representation. Next phase is symbol table which is a data structure used by the compiler to record and collect information about source program constructs like variable names and all of its attributes, which provide information about the storage space occupied by a variable (name, type, and scope of the variables). A symbol table should be designed in an efficient way so that it permits the compiler to locate the record for each token name quickly and to allow rapid transfer of data from the records. Next phase is Error handler: Error handler is invoked whenever any fault occurs in the compilation process of source program. Both the symbol table management and Error handling mechanisms are associated with all phases of the compiler.

When assessing the superiority and maintainability of large C, C++ and Java source code bases, tools are needed for extracting several facts [2] from the source code, such as: Language Support, Pre-Processing, Lexical Analysis, Parsing, Repository and Extracting capabilities of tools.

In this paper, we present our experience in the architecting of Imagix-4D that is a source code analysis tool from Imagix Corporation [3], used primarily for understanding, documenting and evolving existing C, C++ and Java software. Imagix-4D applied technologies include full semantic source analysis. Software visualization supports program comprehension. Static data flow analysis-based verifications detect problems in variable usage, task interactions and concurrency. Software metrics measure design quality and identify potential testing and maintenance issues. The Imagix-4D Reverse Engineering Tool has some inadequacies. That illustrates only abstract Class Diagram which is not easily understood by other developers and users. Imagix-4D does not illustrate ER-Diagram and Sequence Diagram. In this paper we highlight only first inadequacy of Imagix-4D reverse engineering tool.

2. Tool Selection Criteria

In this section we will describe the applied tool selection criteria, the reasons why we have selected particular tools into the study their basic characteristics [4] and Table 1 features extension.

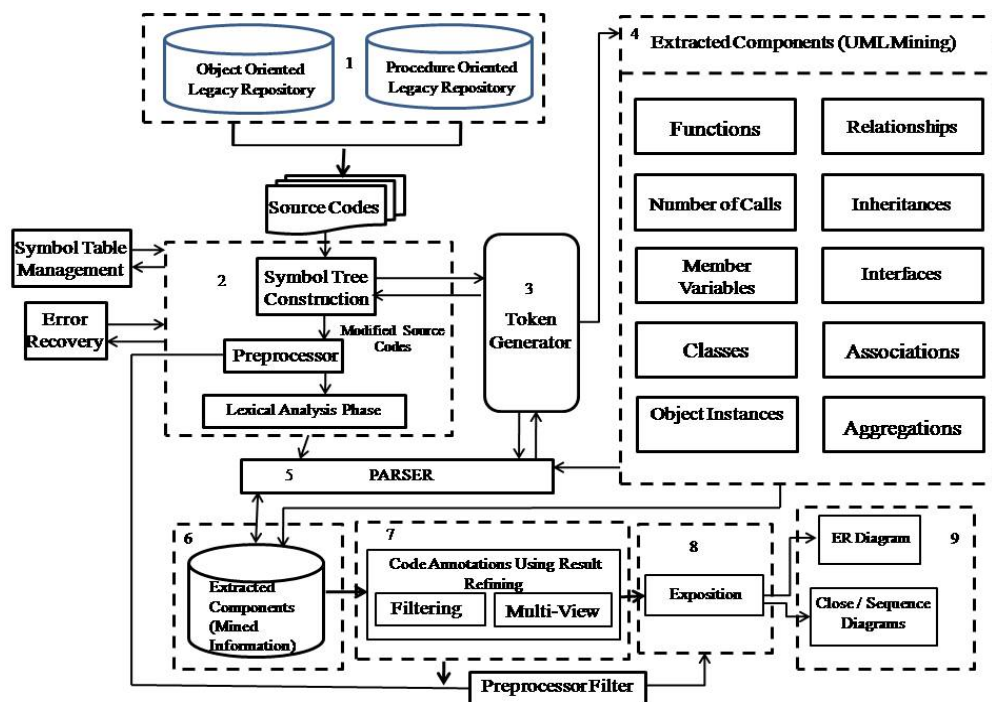


Figure 1. Reverse Engineering Tool based on Unified Mapping Method (RETUM).

Table 1. Behavioral and analytical comparison of existing reverse engineering tools.

RE Tools	Key Attributes	Source Environment	Merits	Demerits
Rigi [5]	Fault-tolerance Completeness Correctness Performance Extensible Scalability Portability Availability Usability	C, C++	1) The major advantages of the tool are that it features new technologies (e.g. layered views, Shrimp view, layout algorithms etc.). 2) The tool provides supporting Capabilities (e.g. filters, metrics, groups, etc.) and it is extensible in some way. 3) The only tool that allows to save the generated views and represent at-ions.	1) The major drawback of Rigi is the provided parser which can only parse functions and structure data Types. 2) This limits the views that can be generated mainly to functional views (call graph). 3) Another Problem is that the tool because it is a research prototype is not too stable.
Doclike Viewer [6]	Performance Scalability Portability Usability	C, C++	1) Doclike Viewer is best to be used within software life cycle.	1) It uses Rigi parser. It does not have its own parser. 1) In view, is limited to that connections can only go in one direction from an entity, therefore, resulting in many representations of one item (e.g. function) if the item is referenced somewhere else. 2) Because of this limitation of the views, highly connected entities cannot be identified and the reading of the views can get complicated with large graphs.
SNIFF+ [7]	Fault-tolerance Scalability Portability Usability	C, C++	1) Sniff+ provides an efficient and portable environment with a comfortable user interface. 2) Sniff+ also provides good printing capabilities. 3) Sniff+ is the only tool that also supports browsing between all generated views which comes in handy sometimes.	1) Adapting SHriMP to new data domains within Eclipse, and applying the idea of terminals to program visualization [9].
Shrimp [8]	Performance Extensible Scalability Portability Availability	C, C++	1) Tool provides a customizable and Interactive environment for navigating and browsing complex information spaces. 2) It employs a fully zoom able interface for exploring software.	1) The visualization will be performed: for every node and edge the user can choose from a selection of metrics is contain little complexity.
Code crawler [10]	Extensible Scalability Portability Availability Usability	C, C++, java small talk	1) It supports reverse engineering through the combination of metrics and software visualization. 2) In this we can see screen shot also.	1) Excludes option for lexical highlighting.
CSV [11]	Correctness Performance Extensible Scalability Portability Availability Usability	C, C++	1) In this user can choose colour for Syntax elements as if statements in C++. 2) Supports gradual zooming up to the point where a line of text becomes 1 pixel.	1) Tool is too generic; needs customized wizards that should address specific questions [13].
Solidsx [12]	Available Portable Usable Scalable Performance Completeness Fault-tolerance	C, C++, .NET/c# and Java code bases	1) It tightly integrates several visual techniques HEBs, tree maps, table lenses with several reverse engineering and analysis in a single environment. 2) The most important feature for user acceptance of Solidsx is integration ease. 3) Solidsx was used in several industrial reverse engineering and program comprehension Projects.	1) It provides low accuracy. 2) Dalli itself cannot extract the complete source code as there is no one tool that can successfully extract the complete source code/ architecture model. 3) Dalli tool required to preprocessing as it allows and analyst to interact with the recovered information by accessing the result of reconstruction effort.
Dalli [14]	Compliance Full coverage Completeness Scalability Portability	Language independence	1) Dalli is recoverable because parsing and lexical technique which is highly versatile. 2) Dalli tool is versatile in light weight then other base technique.	

Continued

GUPRO [15]	Compliance Crossref Completeness Scalability Portability Availability	C, C++, Java, and RDBMS	1) It uses a schema independent querying mechanism. 2) This model implies (conceptual model) the structure of the graph-based GUPRO-repository. Source code is extracted into the repository and the repository graphs can be viewed by an integrated querying and browsing facility. 3) GUPRO has a complete treatment of preprocessor facilities [16].	1) Due to large software system all facts are source cannot fill at once due to Limited repository size, fact extractors for multi-languages systems follow a four step parsing approach [17].
DEFCTO [18]	Fault tolerance Completeness Compliance Crossref Preprocessor Completeness Availability Portability	Language independent	1) Arbitrary factual annotation can be added to the grammar; it is independent from any preconceived analysis model and is fully general. 2) The method is succinct and its notational efficiency has been demonstrated by comparison with other method.	1) This technique does not rely on a specified grammar formalism or parser.
COLUMBS [19]	Fault tolerance Completeness Compliance Crossref Preprocessor Completeness Portability	C/C++ projects and to extract their UML Class Model and call graph	1) It supports project handling, data extraction, data representation and data storage. Furthermore, client entering methods can be used to produce comprehensible (clear-cut) diagrams from the extracted information. 2) Recoverable fault tolerance because data extraction is pre-processed. 3) It is compliance because it is highly adoptable from user as it is a professional tool covering Reverse Engineering Tool in a single package [20].	1) Costly and not ease to availability.
Imagix-4D [7] [3]	Availability Portability Usability Scalability Performance	1) It is used primarily for understanding, documenting and evolving existing C, C++ and Java software. 2) It is also used in Software metrics measure design quality and identify potential testing and maintenance issues.	1) It provides views to rapidly check and systematically study software. 2) Presents key information on software in a 3D-graphical format which enables the user to quickly focus on particular areas of interest. 3) It helps software developers comprehend complex or legacy C, C++ and Java source code. 4) By using Imagix-4D to reverse engineer and analyze our code, we are able to speed your development, enhancement, reuse, and testing. 5) It eliminates bugs due to faulty understanding. 6) It enables us to rapidly check or systematically study your software on any level from its high level architecture to the details of its build, class and function dependencies. 7) We can visually explore a wide range of aspects about your software-control structures, data usage, and inheritance. All based on its precise static analysis of your source. 8) Using this tool we are able to find and focus on the relevant portions of your source code through its querying capabilities. 9) Using this tool we are able to find and focus on the relevant portions of your source code through its querying capabilities [21].	1) The disadvantage of smaller graph is that highly connected graphs get complicated and unreadable. 2) The hand designed class and function diagrams sometimes does not get match with the tool designed diagrams (Class Diagram). 3) The parser lacks of important information about method/function calls which is due to inability of interpreting template parameters (Sequence Diagram). 4) It is unable to resolve the function to which the invocation resolves during compilation time (Sequence Diagram). 5) Imagix-4D requires many hours of analysis for larger code-bases. 6) Imagix-4D does not produce a full executable slice, since it does not perform analysis of relevant conditions for the identified statements. 7) In Imagix-4D has draw Class diagram but it is limited in nature it not give all relationship (Association, Aggregation, Dependencies, Generalization Realization).

Continued

Reveal Tool [22]	Classes, Relationship Dependencies, Generalization, Realization, Aggregation	Input from C++ Code and output as Class Diagram	<ol style="list-style-type: none"> 1) Method based on Keystone. 2) Mechanism used Bottom Up & Backtracking Parse Algorithm Token Decoration. 3) Detection/Mapping attributes based on ambiguity level: Classes it has low ambiguity. 4) Semantically Accuracy in C++ to UML plotting more accurate in and Classes and Association. 5) Ease and sufficient generation of Reverse models. 	<ol style="list-style-type: none"> 1) Detection/Mapping attributes based on ambiguity level: Relationships contains high ambiguity (Dependencies contains high ambiguities, Associations contains high ambiguity, Generalization contains ambiguity, Realization contains medium ambiguity), Aggregation contains high ambiguity.
Rational Rose Tool [23]	Classes, Relationship Dependencies, Associations	Input from C++ Code and output as UML Diagram.	<ol style="list-style-type: none"> 1) Method based on parsing. 2) Mechanism used disassembler. 3) Detection/Mapping attributes based on ambiguity level: Classes it has low ambiguity. 	<ol style="list-style-type: none"> 1) Detection/Mapping attributes based on ambiguity level: Relationships contains high ambiguities (Dependencies contains high ambiguities, and Associations contains high ambiguities). 2) Exact Mapping is not done and less accurate. 3) UML does not include internal dependencies such as method invocations and variable accesses. Those dependencies are necessary in the problem detection and reorganization phases of the re-engineering life cycle. Thus, choosing UML would violate the requirement of being a sufficient basis of re-engineering operations.
Super Womble [24]	Classes	Input from C++ Code and output as Class Diagram.	<ol style="list-style-type: none"> 1) Method based on parsing. 2) Mechanism used Abstract Syntax Tree, Token Stream, Lexical Analyzer. 3) Detection/Mapping attributes based on ambiguity level: Classes it has low ambiguity and Object Diagrams Contains low ambiguity. 	Exact Mapping is not done and less accurate.
Pilfer [25]	Classes, Relations, Dependencies, Association, Generalization, Realization, Aggregation	Input from C++ Code and output as Class Diagram.	<ol style="list-style-type: none"> 1) Method based on parsing. 2) Detection/Mapping attributes based on ambiguity level: Classes it has low ambiguity 3) Light weight Detection. 4) More accurate in graph generation. 	<ol style="list-style-type: none"> Detection/Mapping attributes based on ambiguity level: Relationships contains high ambiguity Dependencies contains high ambiguities, Associations contains high ambiguity, Generalization contains ambiguity, Realization contains medium ambiguity, and Aggregation contains high ambiguity.

Tool Selection Criteria

Because there are numerous tools for reserve engineering purposes it is not possible to analyze all of them in a single study. We have decided to focus on some properties of those tools **Table 1** show details fruition of properties they are: well-known freely available tools which support C, C++ or Java languages. The languages have been selected since they are among both the most commonly used and supported ones. The selected tools should also be either under active current development or be related to scientific publications of software maintenance.

The C programming language is still very important in this context since it is used in numerous important legacy systems which are under maintenance. It is also the only language for which there exist multiple empirical studies on information needs [1] [4]. Object-orientation (OO) is important in the development of new sys-

tems which will be legacy system in the future. The most commonly used OO-languages include C++ and java. Most of the reserve engineering tools support C language. Some of them support also at least some of the OO-languages, most notably C++ or Java, so on bases on above basic properties of the tools we will select Imagix-4D Reverse Engineering Tool environment.

3. Proposed Reverse Engineering Tool Based on Unified Mapping Method (RETUM)

The below architecture in **Figure 1** is proposed for applying reverse engineering on legacy codes of C, C++ class libraries of object oriented or procedure oriented codes. Thus initially the code samples are passed into the code analysis module. This code analysis module takes the code of various languages and makes them separated according to the type of keyword used and store them into a temporary storage. There has been considerable progress in code analysis phase for C, C++, Java, and COBOL. Code analysis phase parse source or intermediate (e.g., byte code) code and produce a database of code entities (e.g., Functions and variables) and relationships (e.g., method invocation, Number of calls, Inheritances, Interfaces, Classes Associations, Aggregations and object instantiation). From here a symbol tree is constructed for correct analysis of tokens according to their uses in codes. Then the work generates various tokens for mapping. These tokens acts as a data extraction components form source codes. In proposed system, there are totally ten components needs to be extracted for accurate mapping of different entity relationships, class and objects instances.

After these components is correctly extracted from UML mining module then a local parse tree is generated and the information is stored in repository for its further usage. Now the direct mapping is possible after this phase but to customize the requirement the proposed work is also adding some more features like code annotation module in which the identified results is further refined by using two specific methods Filtering and Multi-View. This result is then forwarded to exporter which later on plots the identified extracted patterns in a form of Class diagram, Sequence diagram or Call graph as an output.

After analysis it seems that in near future, suggested tool will proves its efficiency and usability in terms of its language supportability (C++/C# and, Java) diagram supportability input range (Class and Activity), detection and mapping mechanism (Various Parameters for accurate mapping). After applying the updated concepts at initial level of work, it is identified that the approach will proves as an unambiguous UML generation from source code and is more accurate, easy and complete.

4. Proposed Algorithm Reverse Engineering Tool

We proposed a algorithm for design Reverse engineering tool of RETUM.

Step 1: First we take legacy codes (object oriented or procedure oriented codes) as input.

Step 2: Legacy code samples are passed into the code analysis module as input. These code analysis modules takes the code of various languages and makes them separated according to the type of keyword used and store them into a temporary storage and symbol tree is constructed for correct analysis of tokens according to their uses in codes.

Step 3: Next step the takes input from code analysis phase and generate token with the help of token generator (generates various tokens for mapping).

Step 4: These tokens acts as a data extraction components form source codes. Extraction components needs to be extracted for accurate mapping from UML mining of different entity relationships, class and objects instances.

Step 5: After these components is correctly extracted from mining module UML mining then a local parse tree is generated and the information is stored in repository for its further usage.

Step 6: Now for the customize the requirement the proposed work is also adding some more features like code annotation module in which the identified results is further refined by using two specific methods Filtering and Multi-View.

Step 7: This result is then forwarded to exporter which later on plots the identified extracted patterns in a form of object oriented diagram or procedure oriented diagram as an output.

We realize of above algorithm for design simplification adaptation of class diagram.

5. Algorithm for Class Diagram Visualizations

Step 1: Initially starts with legacy code or source code as input.

Step 2: Here we take the specific java file as input.

Step3: The UML Doclet API will process the java file (Any additional UmlGraph or *javadoc* arguments can be added at the end of the command line. This command will read the specification file (e.g. Test.java) and generate directly a diagram of the appropriate type).

This option provides the maximum flexibility. In order to run, *javadoc* needs to access tools.jar.

1. Specify the location of tools.jar as a part of Java's classpath and specify the full name of the UML Graph doclet as an argument to Java. This is an invocation example under Windows `java -classpath"lib/UmlGraph.jar; c:\program files\java\jdk 1.6.0_02\lib\Tools.jar" org.umlgraph.doclet.Uml Graph - package Test.java` and under Unix `java -classpath '/usr/share/lib/UmlGraph.jar:/opt/java-1.6/lib/tools.jar'\org.umlgraph.doclet.UmlGraph -package Test.java`
2. Place the UmlGraph.jar file in a directory that also contains the Java SDK tools.jar file.`java -jar /path/to/UmlGraph.jar yourfile 1.java...`

Step 4: The UML graph & UML tool API will extract the relevant data from java file.

`javadoc -docletpath UmlGraph.jar -doclet org.umlgraph.doclet.UmlGraph -private Simple.java`

4.1 Add command line option `umlgen` (generates UML diagrams if the source documentation contains) and `umltypegen` (generates UML diagrams for all documented classes and interfaces).

4.2 Add command line `umlpckagegen` (generates UML diagrams for all documented packages).

4.3 Add command line `umloverviewgen` (generates project overview UML diagrams).

4.4 Add command line `umlautogen` (generates all types of UML diagrams).

Step 5: After step 4, the Maven API is added by UML Doclet.

Step 6: The class diagram is generated and display to the user.

Above algorithm specific used for class diagram generation, which take input as java file and produce output as graphical form details in appendix.

6. Conclusion

In this research paper, we investigate various features of Imagix-4D, and concentrate on class diagram visualization of Imagix-4D. In Imagix-4D class diagram visualization which is more complex, it is not easy to understand a proposed tool RETUM which works on this inadequacy of above tool and illustrates simple comprehensive Class Diagram and we will propose here extension of Imagix-4D Reverse Engineering Tool to draw sequence diagram and ER-Diagram which are Extend Feature of Imagix-4D.

References

- [1] Von Mayrhauser, A. and Vans, A.M. (1995) Industrial Experience with an Integrated Code Comprehension Model. *Software Engineering Journal*, **10**, 171-182. <http://dx.doi.org/10.1049/sej.1995.0023>
- [2] Berndt, B. and Harald, G. (1998) An Evaluation of Reverse Engineering Tool Capabilities. *Journal of Software Maintenance: Research and Practice*, **10**, 305-331.
- [3] <http://www.Imagix.com>
- [4] Yadav, R., Kothari, A. and Patel, R. (2013) Design Generic Architecture for Software Engineering "Extractor" Tool. *International Journal of Scientific & Engineering Research*, **4**, 1490-1495.
- [5] Muller, H.A. and Kienle, H.M. (2010) Rigi—An Environment for Software Reverse Engineering, Exploration, Visualization, and Redocumentation. *Science of Computer Programming*, **75**, 247-263.
- [6] Suleiman, S. (2005) Doclike Viewer: A Software Visualization Tool. *Proceeding of 1st Malaysian Software Engineering Conference (MySEC'05)*, Penang, 12-13 December 2005, 263-265.
- [7] Bellay, B. and Gall, H. (1998) An Evaluation of Reverse Engineering Tool Capabilities. *Journal of Software Maintenance: Research and Practice*, **10**, 305-331. [http://dx.doi.org/10.1002/\(SICI\)1096-908X\(199809/10\)10:5<305::AID-SMR175>3.0.CO;2-7](http://dx.doi.org/10.1002/(SICI)1096-908X(199809/10)10:5<305::AID-SMR175>3.0.CO;2-7)
- [8] Storey M.-A. and Michaud, J. (2001) Shrimp Views: An Interactive Environment for Exploring Multiple Hierarchical Views of a Java Program, in ICSE 2001 (Workshop on Software Visualization).
- [9] Rayside, D., Litoiu, M. and Storey, M.-A. (2001) Integrating SHriMP with the IBM WebSphere Studio Workbench. *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*, Toronto, Ontario, 5-7 November 2001, 79-93.
- [10] Lanza, M. (2003) CodeCrawler—Lessons Learned in Building Software Visualization Tool. *Proceeding of 7th Euro-*

- pean Conference on Software Maintenance and Reengineering, Benevento, 28 March 2003, 409-418.
- [11] Moberts, I.B. (2005) Code Structure Visualization. Master Thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven.
 - [12] Auber, D., Melancon, G., Munzner, T. and Weiskopf, D. (2010) SolidSX: A Visual Analysis Tool for Software Maintenance. Poster Abstracts at Eurographics/ IEEE-VGTC Symposium on Visualization.
 - [13] Reniers, D., Voinea, L., Ersoy, O. and Telea, A. (2014) The Solid Toolset for Software Visual Analytics of Program Structure and Metrics Comprehension: From Research Prototype to Product. *Science of Computer Programming*, **79**, 224-240. <http://dx.doi.org/10.1016/j.scico.2012.05.002>
 - [14] Kazman, R. and Carriere, S.J. (1999) Playing Detective: Reconstructing Software Architecture from Available Evidence. *Journal Automated Software Engineering*, **6**, 107-138.
 - [15] Riediger, V. (2000) Analyzing XFIG with GUPRO. *7th Working Conference on Reverse Engineering*, Brisbane, 23-25 November 2000, 23-25. <http://dx.doi.org/10.1109/WCRE.2000.891466>
 - [16] Ebert, J., Kullbach, B., Riediger, V. and Winter, A. (2002) GUPRO: Generic Understanding of Programs: An Overview. *Electronic Notes in Theoretical Computer Science*, **72**, 47-56.
 - [17] Kienle, H. and Muller, H.A. (2008) 1st International Workshop on Advanced Software Development Tools and Techniques (WASDeTT).
 - [18] Basten, H.J.S. and Klint, P. (2008) Defacto: Language-Parametric Fact Extraction from Source Code SLE. *Lecture Notes in Computer Science*, **5452**, 265-284.
 - [19] Ferenc, R., Beszedes, A., Tarkiainen, M. and Gyimothy, T. (2002) Columbus-Reverse Engineering Tool and Schema for C++. *Proceedings of the 18th International Conference on Software Maintenance*, Timisoara, 12-18 September 2010, 172-181.
 - [20] Boerboom, F.J.A. and Janssen, A.A.M.G. (2006) Fact Extraction, Querying and Visualization of Large C++ Code Bases. Master Thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven.
 - [21] Alexandru, T., Byelas, H. and Voinea, L. (2009) A Framework for Reverse Engineering Large C++ Code Bases. *Electronic Notes in Theoretical Computer Science*, **233**, 143-159.
 - [22] Matzko, S., Clarke, P.J., Gibbs, T.H., Malloy, B.A. and Power, J.F. (2002) Reveal: A Tool to Reverse Engineer Class Diagrams. *Proceeding of the 40th International Conference on Tools Pacific: Objects for Internet, Mobile and Embedded Applications*, Australia, 1 February 2002, 13-21.
 - [23] Schweizer, D., Nierstrasz, O. and Ducasse, S. Exporting MOOSE Models to Rational Rose UML Software Composition Group Institute of Computer Science and Applied Mathematics. <http://www.iam.unibe.ch/dschwzr/>
 - [24] Jackson, D. and Waingold, A. (2001) Lightweight Construction of Object Models from Bytecode. *IEEE Transactions on Software Engineering*, **27**, 156-169. <http://dx.doi.org/10.1109/32.908960>
 - [25] Sutton, A. and Maletic, J.I. (2005) Mappings for Accurately Reverse Engineering UML Class Models from C++. *WCRE'05: Proceedings of the 12th Working Conference on Reverse Engineering*, Washington DC, 7 November 2005, 175-184.

Appendix

Appendix 1: Discussion and Enlightenment of Class Diagram Tool Phase: Class diagrams characterizing the static data and class structure of Java source code. To achieve such a diagrammatic representation, translation rules are defined that transform Java syntax into class diagram.

This diagram is showing **Figure 2** the output of our code when we click on attach button then it give the following output, it must be noted here that this attachment will only accept the .java extension file.

This dialog box will appear **Figure 3** when user will try to click the convert button without selecting the java file. Also this will notify him to add file only after this issue further procedure will be carried out.

This dialog box will appear **Figure 4** only when the attach button is click and by using the browse button user will be able to add the appropriate file *i.e.*, this is simply for choosing a file from documents.

Similarly, this dialog box will appear **Figure 5** when a file is being attached here, after clicking the attach button a java file named vinay.java is uploaded.

Here when we click on convert button after attaching the required file (.java) then we get a dialogue box showing in **Figure 6** that the diagram has been created also with the destination address where the diagram is created in the memory.

```
class Person {
String Name;
public static void main(String a[]){ }
}
class Employee extends Person {
public static void main(String a[]){ }
}
class Client extends Person {
public static void main(String a[]){ }
}
}
```

Standard Class Diagram Generated by RETUM Tool

In Imagix-4D generated class diagram **Figure 7** program doesn't show of internal details of classes' attributes, member function, access mode and data type ,so class diagram generated by Imagix-4D is not understandable by navies developer/user which play crucial role in software quality. Standard class diagram generated by RETUM tool, class diagram which is obtained when the source code named Person.java is made to attach and the converted. This diagram shows in **Figure 8** the relationship among the classes and also it is showing the inheritance relationship between them. It must be noted here that a default constructor has been created here but it was not included in the source code because the program doesn't contain any default constructor but whenever a program is made to run firstly its default constructor gets executed. In this diagram there are three classes which are named as Person, Employee and Client. Among them the Person is the super class and the rest of the classes namely Employee and Client are sub classes. Person class consists of Person() function, where as the Employee class consists of Employee() and main() similarly the client class consist of client() and main() function responsible for its execution.

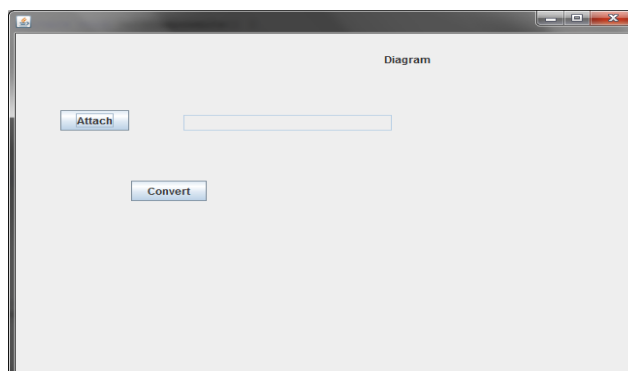


Figure 2. Attachment of the JAVA file.

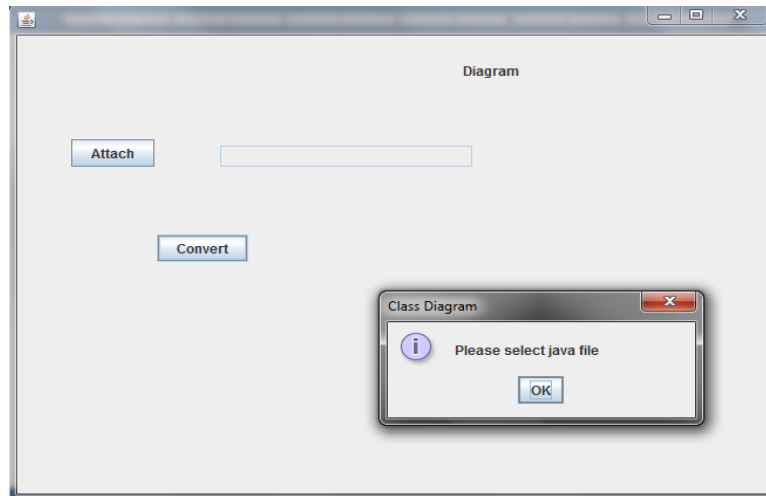


Figure 3. Dialog box alerting to add file.

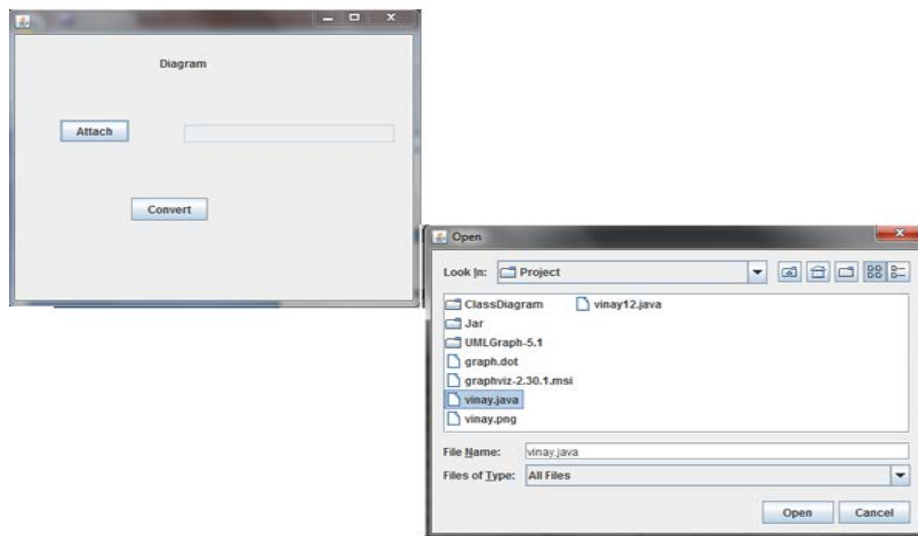


Figure 4. Attach dialog box.

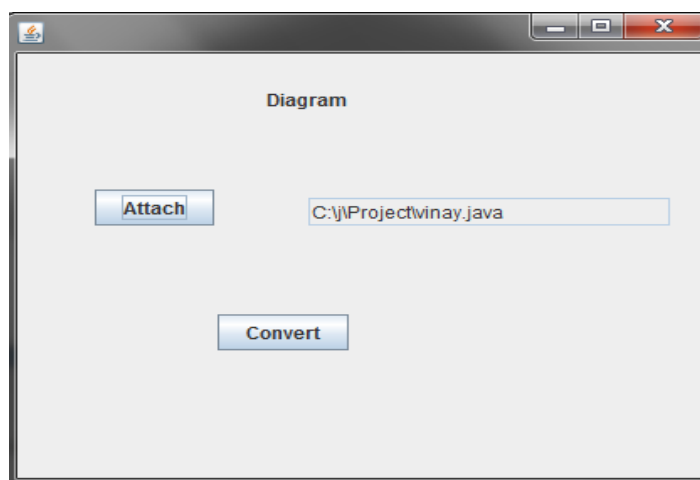


Figure 5. Upload a java file.

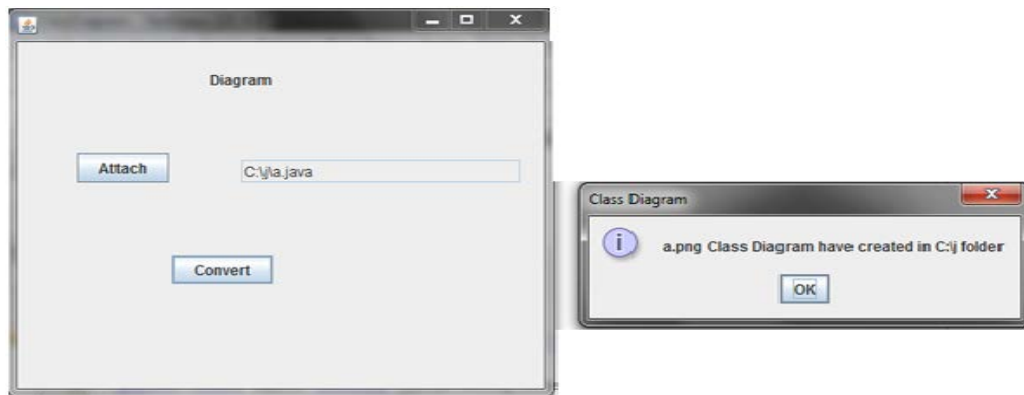


Figure 6. Creation of class diagram.

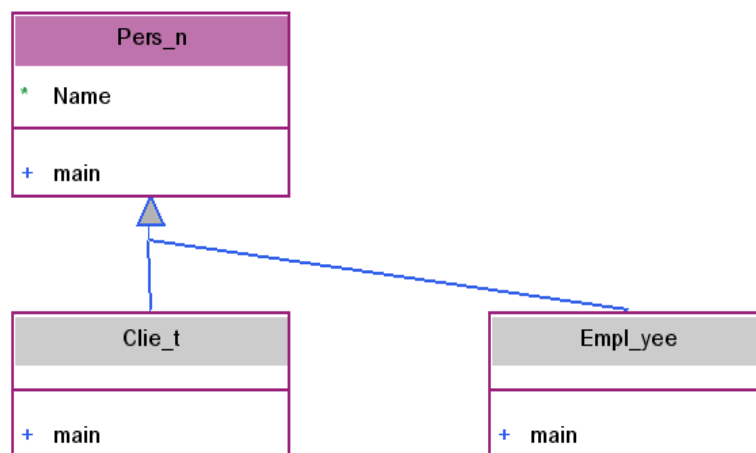


Figure 7. Class diagram generated by Imagix-4D.

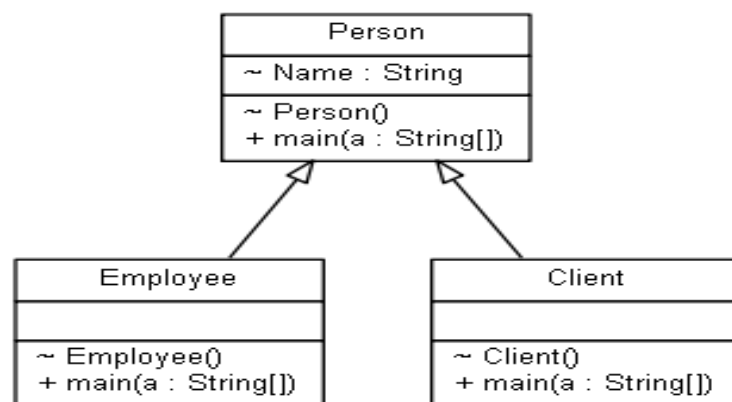


Figure 8. Class diagram generated by RETUM Tool.

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or [Online Submission Portal](#).

