FORWARD AND REVERSE ENGINEERING USING UML WITH RATIONAL ROSE AND OBJECT-ORIENTED PROGRAMMING LANGUAGE

[1]DR G LAKSHMI, [2]DR. KOLLURU SURESH BABU,[3]DR. RATNA RAJU MUKIRI, [4]DR L BHAGYA LAKSHMI, [5]D VENKATA RAVI KUMAR [6]DR SURESH BETAM [7]CHETLA CHANDRA MOHAN

[1]Asst Prof, Dept. of IT, PVP Siddhartha Institute of Technology Vijayawada, AP
[2] Professor & HOD, Dept.of CSE, Vasireddy Venkatadri Institute of Technology, Namburu AP.
[3]Assoc Professor, Dept. of CSE, St. Ann's College of Engineering and Technology, Chirala, AP
[4]Sr Asst Prof, Dept. of Freshman Engineering, Lakireddy Bali Reddy College of Engineering, Mylavaram, AP
[5]Associate Professor, Dept. of CSE, Aditya University, Surampalem AP
[6]Asst Professor Dept. of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, AP
[7]Asst Professor, Dept. of IT, PVP Siddhartha Institute of Technology Vijayawada, AP

Email: chetlachandramohan234@gmail.com

ABSTRACT

Sometimes the source code itself is the only documentation available for post-delivery maintenance. When maintaining legacy systems—that is, software that is still in use but was created no more than 15 or 20 years ago—this occurs far too frequently. It might be quite challenging to maintain the code in these situations. Starting with source code and trying to reproduce the design documents or even the specs is one method of addressing this issue. We refer to this procedure as reverse engineering. This approach can be aided by CASE tools. A nice printer is one of the easiest, and it could make the code easier to see. Other technologies create diagrams, like UML diagrams or flow charts, straight from the source code; these visual aids can support the design recovery process.

One of the two options available to the maintenance team after reconstructing the design is to try to reconstruct the specifications, make the necessary modifications to the reconstructed specifications, and then re-implement the product in the conventional manner. In the context of reverse engineering, forward engineering is the standard development process that moves from analysis through design to implementation. Roundtrip engineering is the process of combining forward and backward engineering. This paper describes how to use UML with Java and Rational Rose to do this.

Keywords*:* *Forward Engineering, Reverse Engineering, UML, JAVA, Rational Rose*

1. INTRODUCTION

Unified Modelling Language is what UML stands for. A language called UML is used to document and visualize the artifacts of software-intensive systems. Model, according to James Rumbaugh, is a simplification of reality. Modelling is the process of capturing key components of a system.

Visual modelling is a type of modelling that uses common graphical symbols. In order to better comprehend the system, we are creating, we construct models.

Models allow us to specify the structure of a system, help us visualize a system as it is or as we wish it to be, and provide a template that directs the system's construction.

Models can serve as a record of our decisions; they can capture business processes; they are a tool for communication; they can manage complexity; and they encourage reuse.

In the late 1980s and early 1990s there were 3 methodologies:

Booch Methodology: This was designed by Grady Booch which is great in design

OMT (Object Modeling Technique) Methodology: This was designed by James Rumbaugh et' al which is great in analysis

OOSE (Objectory) Methodology: This was designed by Ivar Jacobson which is heart of UML i.e., use case

In 1994, James Rumbaugh joined in Rational with Booch and worked together and this is the beginning of unification method. In 1995, Jacobson joined in Rational with Booch and Jim. In 1996, matured unified method was released. In 1997, in January UML 1.0 was released. In 1997 on November 14th UML was accepted by OMG and accepted as a standard language. UML can be used in

1. Banking and Financial services
2. Telecommunications
3.Transportation
4. Defense
5. Retail
6. Modeling Electronics
7.Scientific
8. Distributed Web services

1.1UML Diagrams

Diagram is a graphical representation of elements. UML diagrams can be classified into two types

1. Structural Diagrams 2. Behavioral Diagrams

Structural Diagrams:

These can be divided into 4 types:

i. Class Diagram
ii. Object Diagram
iii. Component
iv. Deployment Diagram

Behavioral Diagrams:

These can be classified into 5 types:

i. Use case diagrams
ii. Activity Diagram
iii. iii. State Chart Diagram
iv. Sequence diagram
v. Collaboration diagram

1.1.1BEHAVIORAL DIAGRAMS

i. **Use case Diagram**:

Use case diagram is created to visualize the interaction of our system with the outside world. The components of use case diagram are:

Use Case: Scenarios of the system

Actor: Someone or something who is interacting with the system

Relationship: Semantic link between use case and actor. The forms of relationship are:

a. Association b. Dependency c. Generalization

ii. **Activity Diagram**

Activity diagram shows the flow of events within our system. The components are:

a) Start State d) Decision Box
b) Synchronization Bar e) End State
c) Transition f) Swim Lane

iii. **Interaction Diagram**

An interaction diagram models the dynamic aspects of the system by showing the relationship among the objects and messages they may dispatch. There are two types of interaction diagrams:

1. **Sequence Diagram**

Sequence diagram shows the step to step what mush happen to accomplish a piece of functionality provided by the system. The components are:

a) Actor d) Lifeline
b) Focus of Control e) Messages f)Object

2. **Collaboration Diagram**

Collaboration diagram displays object interactions organized around objects and their links to one another. The components are:

a) Actor b)Object c)Link

iv. **State chart Diagram**

State chart diagram show a life cycle of a single class. The state is a condition where the object may be in. The components are:

a) Start state
b) State
c) Transition
d) End state

1.1.2STRUCTURAL DIAGRAMS

i. **Class Diagram**

Class diagram shows structure of the software system. The class diagram shows a set of classes, interfaces and their relationships. The components are:

a) Class
b) Relationship:

The forms of relationship are:

1. Association
2. Dependency
3. Composition
4. Generalization 5.Aggregation

ii. **Component Diagram**

Component is a smallest individual physical replaceable part of the system. Component diagram shows the organization and dependencies among software components. The components present are:

a) Component
   a. Runtime component(.dll)
   b. Software components(.h)
   c. Executable components(.exe)
b) Dependency
c) Interface

iii. Object Diagram

Object diagram shows objects and links among objects. The components are:

a) Object b)Link

The object diagram cannot be model in rational rose.

iv. Deployment Diagram

Deployment diagram visualizes distribution of components across the enterprise

2.INTRODUCTION TO RATIONAL ROSE

Rational Rose is a software where the UML can be model. Here, Rational is the name of the software, ROSE stands for Rational Object Software Engineering.

To draw the UML Diagram in Rational Rose:

Step 1: Start Rational software, in that Rational Rose Enterprise Edition. After that Rational Rose Enterprise Edition will be activated. The Rational Rose window contains 5 parts.

1. **View Table**
   It contains:
   a. Use case view c.
   b. Logical view
   c. Component view
   d. Deployment view
2. **Diagram Tool Bar**
   This can contain the tools of the corresponding diagram in which we are going to draw
3. **Diagram Window**
   In this window we can draw the diagram
4. **Message Window**
   It contains the message of documentation of the corresponding diagram
5. **Log Window**
   This is the place where the errors can be displayed when we are drawing the diagram

2.1 USECASE VIEW

In this view we can draw two diagrams:

1. Use case diagram
2. Activity Diagram

Steps to draw diagram:

1. Select use case view and then right click on use case view
2. Select New, in that select use case/activity diagram [
3. Name the diagram
4. After double clicking on the diagram name, the corresponding use case/activity will be opened
5. We can draw the diagram by drag and drop the components of the corresponding diagram

2.2LOGICAL VIEW

In this view we can model:

a. Class diagram
b. Sequence diagram
c. Collaboration diagram
d. State Chart diagram

To draw the diagram:

1. Select logical view and then right click on the logical view
2. Select new in that select class/ state chart/ sequence diagram
3. Name the diagram
4. After double clicking on the diagram name, the corresponding diagram will be opened
5. We can draw the diagram by drag and drop the components of the corresponding diagram

2.3 COMPONENT VIEW

In this view we can model Component Diagram

To draw the diagram:

1. Select component view and then click on the component view
2. Select New, in that select component diagram
3. Name the diagram
4. After double clicking on the diagram, the corresponding diagram will be opened
5. We can draw diagram by drag and drop the components of corresponding diagrams

2.4 DEPLOYMENT VIEW

In this we can model deployment diagram

To draw diagram:

1. Select deployment view, then right click on deployment view
2. Select New, in that select deployment diagram
3. Name the diagram
4. After double clicking on the diagram name, the corresponding diagram will be opened
5. We can draw diagram by drag and drop the components of corresponding diagrams

3.UNIFIED LIBRARY APPLICATION(ULAS) INTRODUCTION

The Unified Library Application System places a strong emphasis on online book reservations, loan, and return. The current library system is made global by this system. The member can reserve any book from anywhere in the world by using this application. This application, which is still in its infancy, will soon transform the current library system.

A brief synopsis of the unified library application system is as follows:

- Librarian lends books and magazines and maintains list of members.
- Librarian maintains the list of all the members of library
- Borrower makes reservation online
- Borrower can remove reservation online
- Librarian issues books to the borrower and calculate bills.
- Borrower issues/returns books and/or magazines
- Librarian places order about the requirements to the master librarian
- Librarian updates system
- Master librarian maintains librarians

3.1TEXTUAL ANALYSIS

(a) ACTORS

i. Librarian
iii.Catalog
ii. MasterLibrarian
iv.Borrower

(b) VERBS

i. Borrower:

1. Logs into the system
2. Browses/searches for books or magazines
3. Makes/removes reservation
4. Views results and reports from the unified library application system

ii. Librarian:

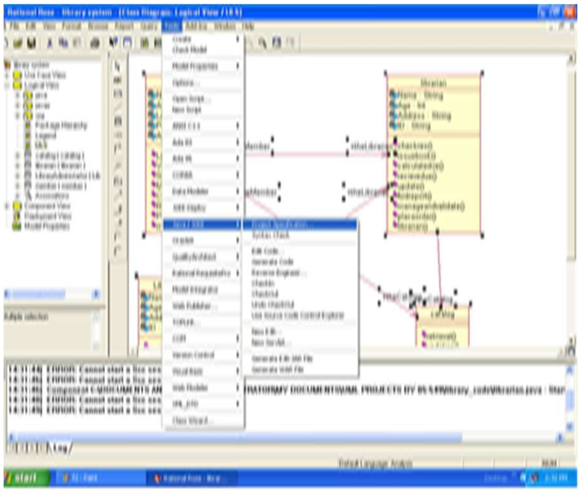
1. Manages and validates members
2. View reports from the system
3. Issues books
4. Calculates dues
5. Takes books
6. Places orders to the master librarian
7. Maintains list of books and magazine

iii. Master Librarian

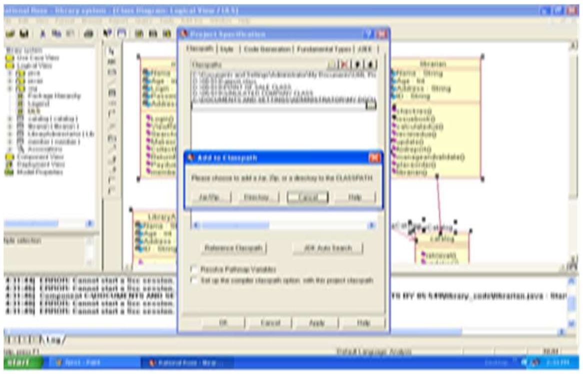
1. Maintains other librarians

3.2FORWARD ENGINEERING: steps to do the forward engineering by considering ULAS as Case Study

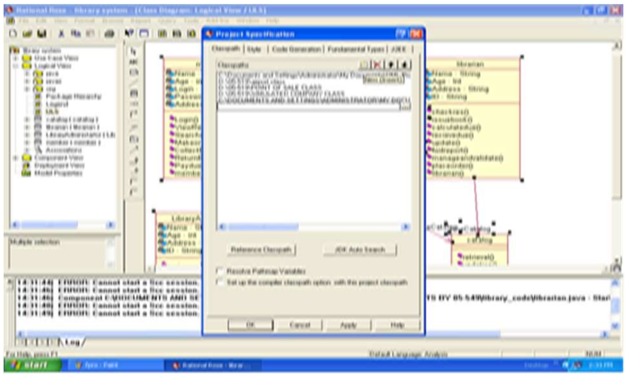
a. Step1 – Project Specification



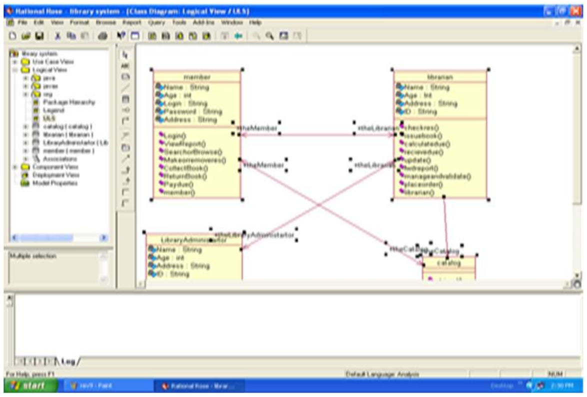
b.Step2 – Set Path



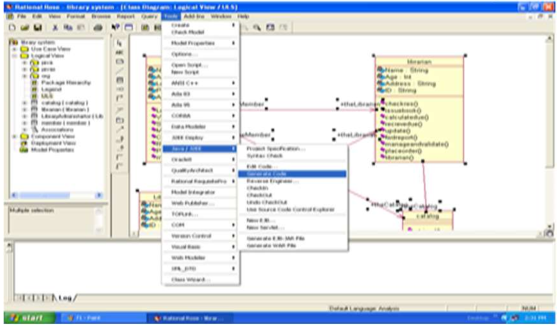
Step3:-.Browse the path



Step4 – Select the classes to be forward engineered



step5-Select Tools -> Java/J2EE -> Generate ccode to forward engineer



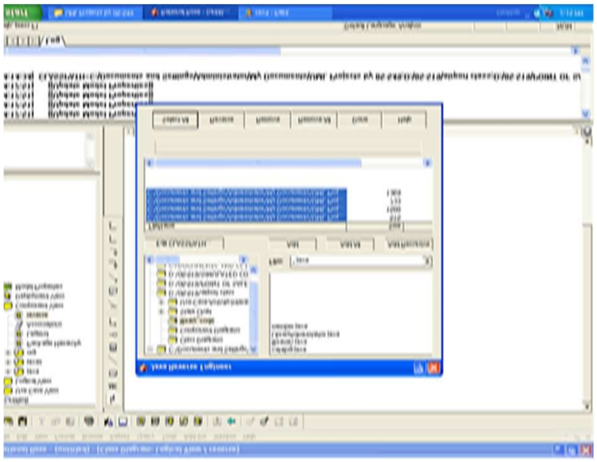
The code is generated at the specified path that is specified under Project Specification part as:

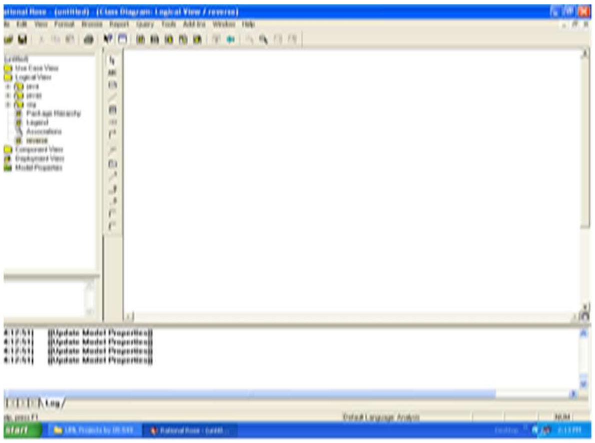
Administrator:

```
public class administrator {
   private String name;
   private String ID;
   public Librarian theLibrarian;
   public administrator() {   }
   public String receive_order() {
    return null;}
   public void manage_librarians() {}
public void purchase_new_stock() {}}
```

3.3REVERSE ENGINEERING: steps in reverse engineering

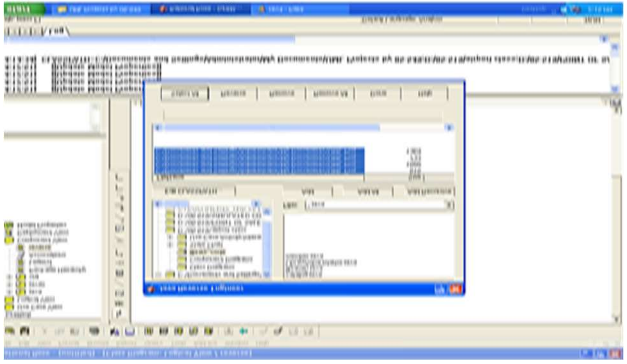
Step 1 :- Open new project -> Class Diagram under Logical View




Rational Rose - (untitled) - [Class Diagram: Logical View / reverse]
[Update Model Properties]
[Update Model Properties]
[Update Model Properties]
[Update Model Properties]
Log
Default Language: Analysis

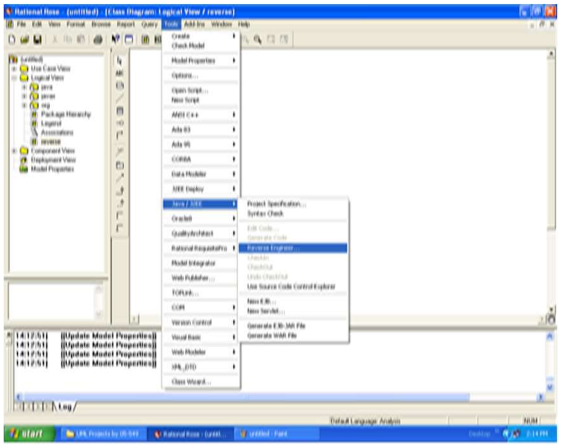
Step2 :-Select Tools -> Java/J2EE -> Reverse Engineer to reverse engineer the code

Step2 :-Select Tools -> Java/J2EE -> Reverse Engineer to reverse engineer the code



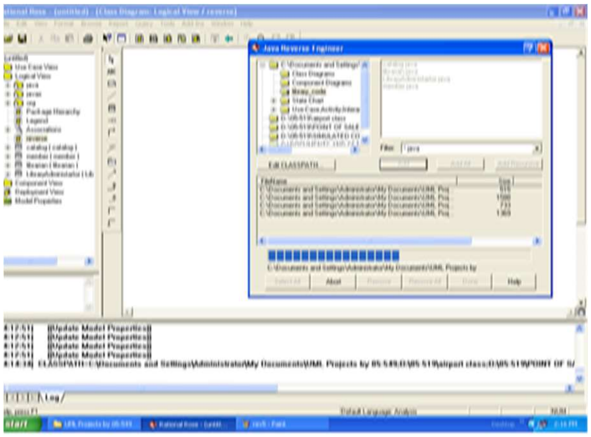
Sstep3 :-Specify the path of code to reverse engineer. Select the files to be reverse engineered

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195


Rational Rose - (untitled) - [Class Diagram: Logical View / reverse]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
Java / J2EE
Project Specification...
Syntax Check
Reverse Engineer...
Generate EJB-JAR File
Generate WAR File

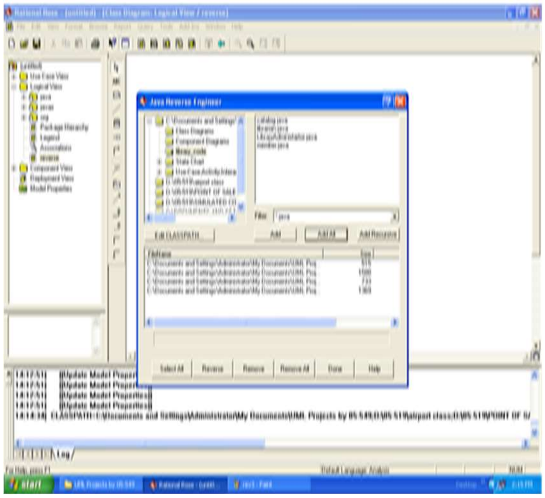
```
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
```

Sstep4 :-After adding all code files click Select All and then click Reverse.

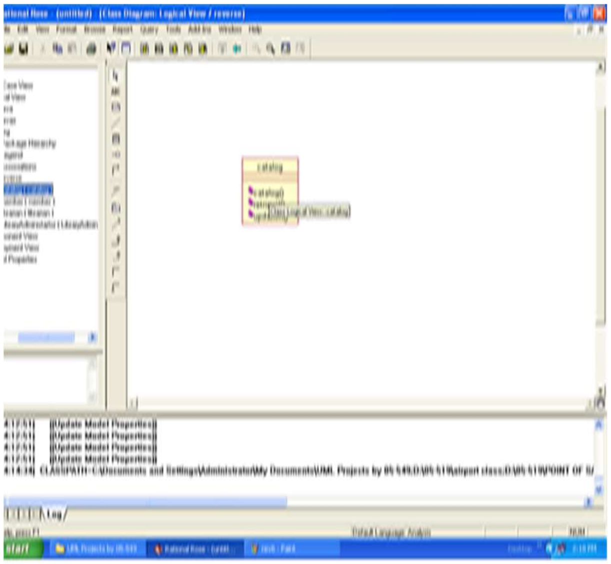


```
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:14:34| CLASSPATH=C:\Documents and Settings\Administrator\My Documents\UML Projects by 05 549;D:\05 51\airport class;D:\05 51\POINT OF S/
```

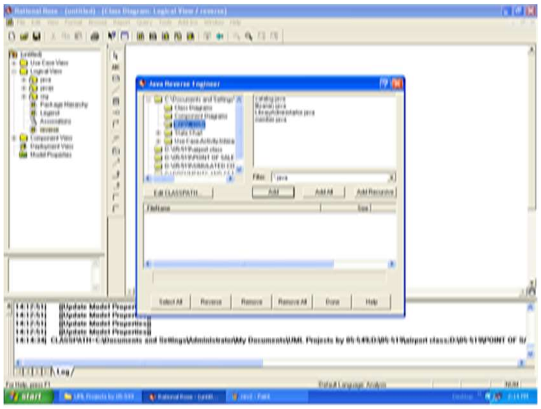
. step 5.The classes generated in the tree window. Drag all the classes to the required area. The associations among the classes is generated automatically



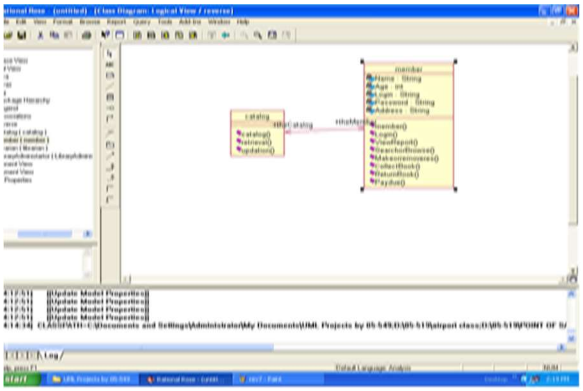
```
14:12:51|    [Update Model Proper
14:12:51|    [Update Model Proper
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:14:34| CLASSPATH=C:\Documents and Settings\Administrator\My Documents\UML Projects by 05 549;D:\05 51\airport class;D:\05 51\POINT OF S/
```



```
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:14:34| CLASSPATH=C:\Documents and Settings\Administrator\My Documents\UML Projects by 05 549;D:\05 51\airport class;D:\05 51\POINT OF S/
```



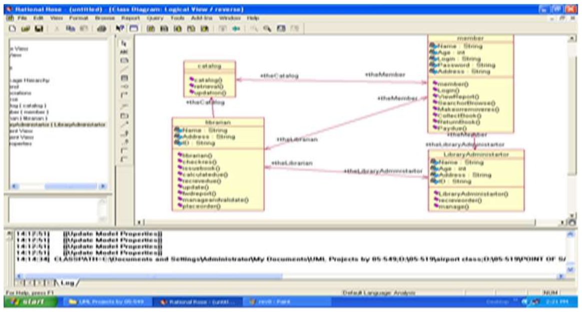
```
14:12:51|    [Update Model Proper
14:12:51|    [Update Model Proper
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:14:34| CLASSPATH=C:\Documents and Settings\Administrator\My Documents\UML Projects by 05 549;D:\05 51\airport class;D:\05 51\POINT OF S/
```



```
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:12:51|    [Update Model Properties]
14:14:34| CLASSPATH=C:\Documents and Settings\Administrator\My Documents\UML Projects by 05 549;D:\05 51\airport class;D:\05 51\POINT OF S/
```

at last the reverse engineering is complete.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



```
14:12:51|     [[Update Model Properties]]
14:12:51|     [[Update Model Properties]]
14:12:51|     [[Update Model Properties]]
14:12:51|     [[Update Model Properties]]
14:14:34|  CLASSPATH=C:\Documents and Settings\Administrator\My Documents\UML Projects by 05-549;D:\05-519\airport class;D:\05-519\POINT OF S/
```

4. CONCLUSION: Forward and reverse engineering are combined to create roundtrip engineering. This paper describes how to use forward engineering to produce the skeleton source code from the comprehensive design and reverse engineering in UML with Rational Rose and Java to extract visual notations such as UML diagrams from the documentation called source code.

REFERENCE:

[1].Pankaj Jalote ," An Integrated Approach to Soft ware Engineering" 2 nd Edition , Narosa Publishing House,2004,Chapter-4 (Planning a Software Project),Pg no. 166-170. ISBN – 81-7319-271-5

[2].Roger S Pressmen, "Software Engineering - a Practitioner's Approch" 6th Eddition Mc Graw Hill international Edition, Pearson education, ISBN 007 - 124083 - 7

[3]Waman S Jawdekar, "Software Engineering Principles and Practices" Tata Mc graw hill ISBN 0 -07 - 058371 – 4

[4] Walker Royce "Software Project Management - A unified frame work" 2nd Edition, low price Edition, ISBN 81 - 7758 - 378 - 6, pearson education

[5].Ian somarville, "Software Engineering" 5th Edition low price Edition, International Computer Science Series.

6. Shari, Laurance, Pfleeger, "Software Engineering theory and practies" 2nd Edition, ISBN 81 - 7808 - 4589 - 7, low price edition.

[7].Carlo ghezzi, Mehdi Jazayeri Dino Mandrioli, "Fundamentals of Software Engineering", PHI, ISBN 81 - 203 – 0865.

[8]Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The unified modeling language user guide* (2nd ed.). Addison-Wesley. ISBN: 978-0321267979

[9]Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language* (3rd ed.). Addison-Wesley. ISBN: 978-0321193681

[10]Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *The unified modeling language reference manual* (2nd ed.). Addison-Wesley. ISBN: 978-0321245625

[11]Sommerville, I. (2015). *Software engineering* (10th ed.). Pearson. ISBN: 978-0133943030

[12]Dennis, A., Wixom, B. H., & Tegarden, D. (2020). *Systems analysis and design: An object-oriented approach with UML* (6th ed.). Wiley. ISBN: 978-1119803784

[13]Arlow, J., & Neustadt, I. (2005). *UML 2 and the unified process: Practical object-oriented analysis and design* (2nd ed.). Addison-Wesley. ISBN: 978-0321321275

[14] Larman, C. (2004). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development* (3rd ed.). Pearson. ISBN: 978-0131489066

[15]Whitten, J. L., & Bentley, L. D. (2007). *Systems analysis and design methods* (7th ed.). McGraw-Hill. ISBN: 978-0073052335

[16]Object Management Group (OMG). (2017). *OMG unified modeling language (OMG UML), version 2.5.1.* Retrieved from https://www.omg.org/spec/UML/2.5.1

[17]Roff, T. (2003). *UML: A beginner's guide*. McGraw-Hill. ISBN: 978-0072224603