RTET - A Round Trip Engineering Tool

Leckraj Nagowah, Zarah Goolfee and Chris Bergue
Computer Science & Engineering Department,
University of Mauritius, Réduit, Mauritius
Email: l.nagowah@uom.ac.mu, {bibi.goolfee, jean-maurice.bergue}@umail.uom.ac.mu,

***Abstract*—Generating codes from models and performing round trip engineering is a key concern in software development. It is vital that software related artifacts such as source codes and models remain in synchronization throughout the development process. Although there are a number of round trip engineering tools available, only a few of them have been adopted by the developers' community. The existing tools perform considerable round trip engineering, but, even then, developers have still lots of work to do to implement a complete system. Many of the developers would like to be able to automatically generate a deployable web application from a model and/or reverse engineer their source codes into models in only a few mouse clicks. This paper introduces our Round Trip Engineering Tool-RTET which follows the Model Driven Engineering (MDE) paradigm to generate a CRUD oriented application and performs reverse engineering to better meet requirements of developers in Java. The prototype is able to automatically generate a working version of a tiered application with a JSP presentation, EJB manager classes with built in functions in Java, and with an appropriate database model. RTET is also able to generate MVC web applications with JSF views and appropriate managed beans. These files are derived from an entity bean which itself is derived from a simple class diagram using eUML2 plug-in. From an existing user interface in JSP or JSF, RTET can also reverse engineer the page to generate an EJB manager class with all its CRUD functions and the corresponding entity bean.**

Keywords— Round Trip Engineering; Code Generator; Forward Engineering; Reverse Engineering; Java Server Pages; JSP; Java Server Faces; JSF; EJB; Beans; Session; Managed; Entity; Controller; Manager Class; Java EE.

I. INTRODUCTION

The Model Driven Engineering (MDE) focuses software development on models rather than on code. A model of a system is a description or specification of that system and its environment for some certain purpose [1]. A model is often presented as a combination of drawings and text. Round trip engineering is a common term used to generate codes from models and keep the models and codes in synchronization with each other. Unified Modeling Language (UML) is a widely used modeling language to represent software requirements into models that give a better overview of the software to be developed. Designing and managing software is a crucial activity during software development. Thus, round trip engineering tools enable developers to focus on complex issues rather than spending most of the time on repetitive coding. Round trip engineering tools ensure good and consistent code generation and perform convenient reverse engineering. It hence consists of forward and reverse engineering. Forward engineering covers the aspects of converting the UML diagrams/models into relevant codes and reverse engineering synchronizes the codes with the models. The synchronization ensures that the models are accurate and relevant as the code changes.

Automatic code generation is very beneficial to software developers since it saves developers time during the development of the software and also makes sure that the codes are generated according to existing templates. While many round trip engineering tools are available on the market, people tend to move towards those that provide for more advanced features and which really help them in the development lifecycle. Most round trip engineering tools generate skeleton codes from UML diagrams. Those codes might merely consist of set and get accessors of the attributes. However, the generation of such codes is not of much use to software developers since any Java IDE such as Eclipse and NetBeans can perform such tasks.

This paper highlights the development of an easy-to-use round trip engineering tool, RTET that generates codes and performs notable reverse engineering of the codes. The tool automatically generates a tiered EJB system with JSPs/Servlets [2] implementing the user interface, EJB session and entity beans for the business logic functions and the data definitions for the tables for a selected database. MVC web applications can also be generated with JSF views [3] and appropriate managed beans. RTET can also reverse engineer a JSP or JSF user interface into a complete EJB session and its entity beans.

II. RELATED WORKS

E. Palacios-González et al. [4] provide an extensive review of available tools that follow the Model Driven Engineering (MDE) paradigm. From the point of view of E. Palacios-Gonzalez et al [5], Borland Together, AndroMDA and Eclipse Modeling tools are very nice options for tools that support MDE. In this section, an overview of some of the tools is presented.

The popular round trip engineering tool, Borland Together [6], implements the basic ideas of round trip engineering concepts. Together provides editors to view both the codes and the UML diagrams at the same time so as to better view the synchronization of the codes and the models. Together supports for languages such as Java, C# and Visual Basic. However, Borland Together does not provide for code generations beyond this and is also not an open source software.

978-1-4673-4992-5/13/$31.00 ©2013 IEEE

Rational Rose [7] is another remarkable round trip engineering tool. It provides for most features of round trip engineering. It supports business modeling and helps with systems analysis by allowing the design of Use Case diagrams to illustrate the system functionality. Rational Rose offers a set of visual modeling tools for development of robust, efficient solutions to real business needs in the client or server, distributed enterprise, and real-time systems environments. Although the tool has numerous benefits, code generation with Rational Rose seems to be limited to classes only. Furthermore, it is proprietary software and does not integrate easily with open source IDEs to readily benefit from its available features.

ArgoUML [8] is an open source round trip engineering tool that supports Java, C++ and PHP languages. Other languages can be supported through its modular framework. Since it has been programmed in Java, ArgoUML is platform independent. Despite its numerous benefits, it does have some drawbacks. Code generation for the Java language also seems to be limited to classes. Moreover the in-built templates for code generation are not accessible by the users and hence cannot be modified. ArgoUML also does not allow viewing the synchronization of codes and models [9].

Gentleware AG Poseidon For UML [10] is a tool that is based on ArgoUML. It supports code generation for languages such as Java, C++, C#, PHP and Visual Basic. However, similar to ArgoUML, code generation seems to be limited to classes only.

JBoss Seam [11, 12] is an open source application framework developed for Enterprise Java. Seam has a standard component model for all business logic in an application. Seam integrates JSF with EJB 3.0 along with AJAX. It uses command line utility to generate some skeleton Seam codes and performs reverse engineer of an application from a preexisting database but not from an interface.

The Spring Web MVC framework [13, 14] is an open source framework for developing rapid web applications using the MVC design pattern. It binds business object directly to HTML forms fields. In resemblance to Seam framework, Spring framework also makes use of command line to perform its functionalities. Spring uses JSP as a view technology. It is a lightweight framework enabling construction of enterprise-ready applications based on POJO (Plain Old Java Objects). Spring only performs reverse engineering of database tables to Hibernate objects.

AndroMDA [15] is an open source code generation framework that follows the Model Driven Architecture (MDA) paradigm. It can generate deployable Java EE applications from a UML model using Hibernate, EJB, Spring and Structs frameworks. However, it does not support reverse engineering. If an application needs to be modified, the UML model should be changed and the code regenerated.

Other similar tools such as ArcStyler and OptimalJ have also existed but have been discontinued [5].

While the above tools focus on true round-trip engineering, i.e. generating source codes from UML models and vice versa by automatically generating source code stubs and automatically converting the changes in the source code into UML model elements, they do not bring much benefit to the developers. The latter would rather enjoy having a complete system that implements the CRUD (Create-Read-Update-Delete) functions for a particular UML model. If some tools are able to generate a deployable system, this is only possible through a number of plug-ins. Many of these tools however provide only for code generation and are not able to fully reverse engineer an application.

III. SYSTEM DESIGN

The challenges to be considered while developing a round trip engineering tool are very critical to high performance, good coding conventions, reliable and maintainable codes. This section highlights some important design issues and describes the main components of RTET. The system is initially based on eUML2 Free Edition Plug-in in Eclipse IDE to design class diagrams and obtain the generated EJB entity class of the UML class diagram. The capabilities of our system, RTET, start at the end of most available tools. The prerequisite of such a system is to have a user friendly graphical user interface which shall accomplish the tasks of RTET in a number of mouse clicks. The GUI shall allow users to choose appropriate parameters such as the template to use, the functions to add in the EJB and the choice of database among others.

The system generated by RTET would need to be a tiered EJB system with JSPs/Servlets responsible for the user interface; session beans and entity beans which implement business logic functions and rules; and the necessary tables.

Information that can be easily identified from a class diagram is the name of the class, the attributes of the class and the methods or operations the class can take or undertake. This information is used by the tools to generate Classes, e.g. in Java or C# and vice-versa.

Our tool RTET extends this functionality to web applications. The environment should enable users to browse for the entity class already generated by any third party system to perform activities designed by the system. The entity bean is parsed by RTET to extract the name and attributes of the class. Based on the user choice of parameters, an EJB manager class is generated with functions such as Add, Delete, Update, Search and FindAll. Consequently, different web pages are also generated which eventually makes use of the functions of the EJB. RTET shall also be able to reverse engineer a JSP or JSF page into its corresponding EJB class and generate the corresponding entity bean.

Fig. 1 shows an overview of RTET and its main components and Table I gives a description of the main tasks carried out by RTET. The code generation process is performed in 2 main steps as described in Table II and Table III. Table II shows the generation of the EJB session beans implementing the CRUD and viewAll functions, and Table III highlights the generation of the user interfaces in JSPs. The system can also generate a full MVC JSF system with appropriate controller class (managed beans) and appropriate views in JSF.

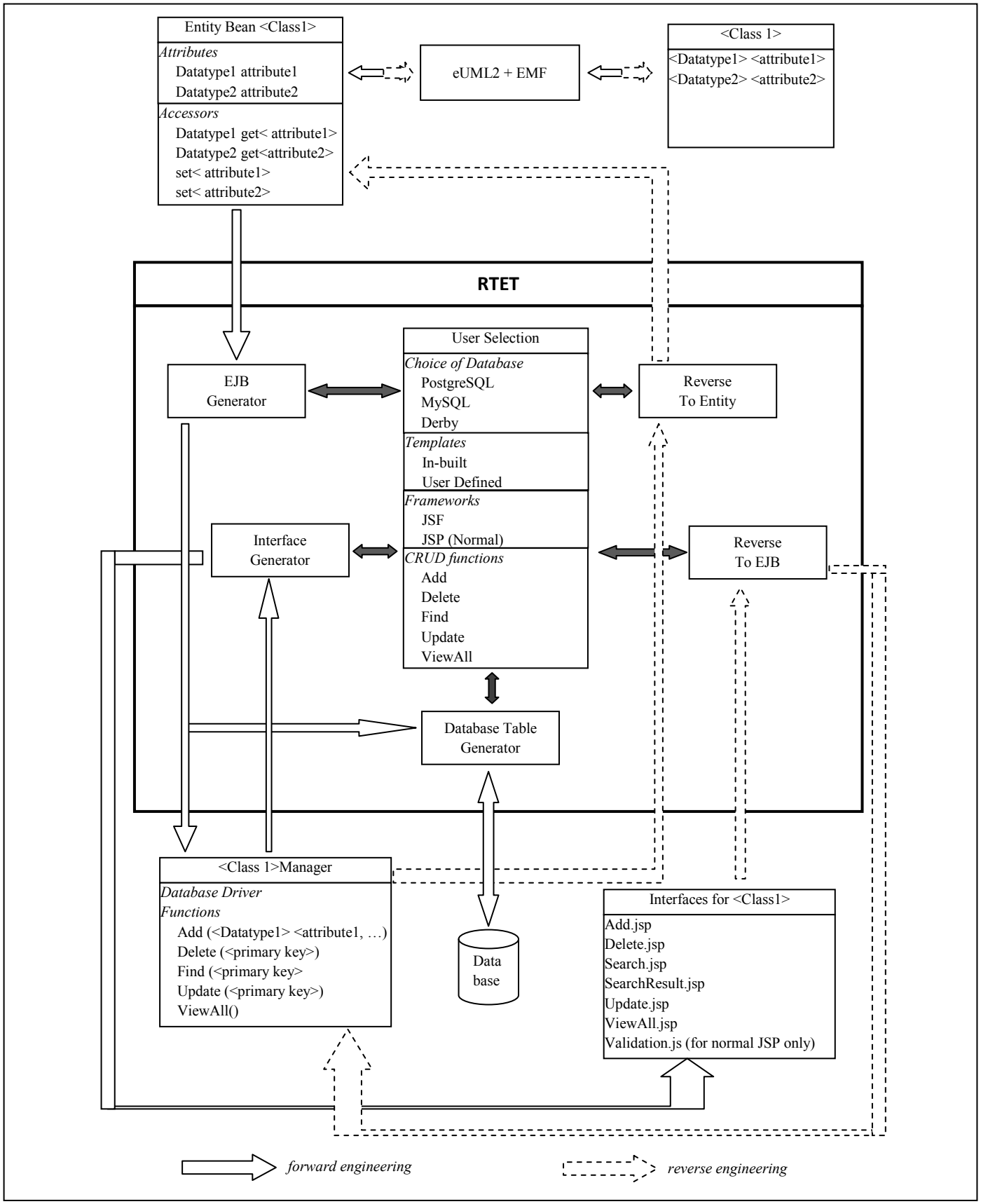

Figure 1. Overview of RTET

TABLE I. SYSTEM DESIGN

1. Generate EJB class based on entity class and database chosen by user.
Should allow for modification of entity details to be applied to EJB class
EJB class should consist of following methods:
Add, Delete, Update, Search, FindAll
These methods are CRUD methods implemented to be able to interact with the database. However the user must be able to select/deselect the methods to be generated in the EJB class.
A table should automatically be created in the chosen database so as to pair with the CRUD functions.
2. Flexibility of generating template codes in EJB class for a specific method instead of the standard codes provided by the system.
User can choose another code format either from the inbuilt templates provided by the system or by providing their own code samples.
The template codes provided by the user should be able to remain permanently in the system for future use unless an administrator decides to remove it.
The template codes may consist of codes related to other frameworks such as Hibernate, Spring, Struts etc....
3. Generate interfaces to be able to interact with the CRUD functions generated in the EJB class and the database.
User should be able to modify details of EJB class in real time to be reflected in the interfaces to be generated.
User should be able to select among the interfaces to be generated.
The interfaces should be of either normal JSP pages or JSF framework, as per choice of user.
Interfaces generated for both frameworks are the Main Menu interface that will consist of links to the Add, Delete, Update, Search, Viewall and errorDisplay interfaces. However, JSP framework should also generate a Validation JavaScript for validation purpose of the interfaces, where as the JSF framework should generate a Controller class with Faces-config.xml and web.xml pages.
4. Reverse engineer the interfaces back to EJB class.
The system shall reverse any interface page of any framework to generate its corresponding EJB class.
5. Reverse engineer EJB session class to the EJB entity class which shall be synchronized back to the UML class diagram.
The system shall accept any EJB session bean and reverse engineer the class back to a corresponding entity class.

TABLE II. EJBGENERATOR WITH TEMPLATE INTEGRATION

Step 1: Browse entity java file generated from UML class diagram created using eUML2 plug-in
Step 2: Select database at client side from combo box among PostgreSQL, MySQL or Derby Db
Step 3: Browse and select existing template for EJB generation or Modify existing template if needed to create a new one. Save the edited code (if any) in a template
Step 4: Process the entity bean and the selected template to generate EJB session bean
If method retrieved from entity class has add() method
Call toAdd() – to generate method add() in the EJB class
If method retrieved from entity class has delete() method
Call toDelete() – to generate method delete() in the EJB class
If method retrieved from entity class has update() method
Call toModify() – to generate method update() in the EJB class
If method retrieved from entity class has viewall() method
Call toView() – to generate method viewall()in the EJB class
If method retrieved from entity class has search() method
Call toSearch() – to generate search() in the EJB class
Create the table in the database
Table Name = Name of Entity Class
Table Fields = Attributes of the class
Data types of Fields = Data types of attributes
End

TABLE III. INTERFACE GENERATOR

Step 1: Browse EJB class
Step 2: Modify existing EJB if needed
Step 3: Select framework for the interfaces
Step 4: Select background color for interfaces
Step 5: Process EJB Class to identify existing functions
If Framework chosen is JSP (Java Server Pages)
Generate MainMenu.jsp, errorDisplay.jsp and validations.js pages
Else Framework is JSF (Java Server Framework)
Generate Main.jsp, controller class and update the XML files of the project
If EJB has add() method
Generate Add.jsp page
If EJB has delete() method
Generate Delete.jsp page
If EJB has update() method
If framework is JSP
Generate Update.jsp page
Else if framework is JSF
Generate Update.jsp and Updated.jsp
If EJB has viewall() method
Generate ViewAll.jsp page
If EJB has search() method
If framework is JSP
Generate Search.jsp and SearchResult.jsp page
Else if framework is JSF
Generate Search.jsp
End of Interface Generation

As shown in Fig. 1, RTET consists of the following main components that make the business logic of our roundtrip engineering tool RTET: EJB Generator, Table Generator, Template, Reverse to EJB and reverse to Entity.

- EJB Generator

The EJB Generator component aims at generating EJB classes based on entity classes obtained from the class diagram using eUML2 plug-in. The generated EJB class generally consists of implementing CRUD (Create, Read, Update, and Delete) methods based on the user preference. It also implements a viewAll function that returns a list of all the entities from the database. The EJB Generator extracts the names and the data types of the attributes from the entity bean. It also gets the preferences of the user in terms of the choice of database, framework to be used for the code generation and the functions to be included in the EJB.

- Table Generator

This component is used by the EJB Generator to automatically create a table in the selected database so as to interact with the CRUD functions or business logic implemented in the EJB class. The table is named after the entity class and the table fields have the name and data types as the attributes of the entity class. Currently a choice of 3 databases is supported by RTET which are PostgreSQL, MySQL or Derby database.

- Template

A Template module is used by the EJB Generator component to provide for flexibility in the generated codes. This module contains inbuilt template codes for the different CRUD functions that are stored in the system database. Codes of any framework or any programming style can also be

provided as templates in RTET to perform code generation according to the specific framework or programming style.

- Interface Generator

This component generates interfaces based on the EJB class and chosen framework. It parses the EJB session bean to identify existing functions and attempts to generate an interface for each of these functions. A choice of JSP and JSF frameworks is currently supported by RTET. Based on the framework, appropriate interfaces are generated. If normal JSP is chosen, interfaces are generated according to the CRUD functions in the EJB class, together with an appropriate validation JavaScript page to validate relevant fields in the generated interface pages. Regarding JSF framework, along with the interface pages for CRUD functions, the corresponding controller class is generated while updating the Faces-config.xml and web.xml file in the project.

- Reverse to EJB

This component reverse engineers any interface of any framework to the corresponding EJB class. The Java libraries scan and retrieve the names of the entity and its corresponding attributes from the interface and generate a corresponding EJB session bean with necessary CRUD functions.

- Reverse to Entity

The Reverse to Entity component reverse engineers an EJB class to its appropriate entity class. Applying the same principles, relevant information from the EJB session bean are read and retrieved so as to generate the entity class.

IV. SYSTEM IMPLEMENTATION & TESTING

Focusing on the design issues, an initial version of RTET has been developed. Rather than enhancing existing round trip engineering tools, RTET has been developed from scratch and provides for innovative functionalities compared to other round trip engineering tools. Moreover, RTET makes use of eUML2 Free Edition plug-in for modeling the UML class diagrams and generate related entity class. The system is deployed and tested on Windows and makes use of the GlassFish server [16] to deploy the web application. Indigo version of Eclipse IDE has been used to implement the system further embedding important API such as Eclipse Modeling Framework (EMF) [17,18], Graphical Editing Framework (GMF) and Ecore to be able to use the eUML2 plug-in, supporting UML 2.0 features.

The Round Trip Engineering Tool, RTET, consists of multiple layers that interact differently with each other, thus supporting an N-tier architecture. The Presentation Layer of RTET is made of up SWING and AWT components. The business logics of RTET which consists of the components illustrated in Fig. 1 and discussed in the previous section, has been implemented in Java. PostgreSQL database has been used to store the preferences of the user and the different templates of RTET.

A prototype application has been used to test RTET. This prototype consists of an Employee Information System, EIS. The EIS is modeled into a class diagram using the eUML2 plug-in in Eclipse IDE. The class diagram consists of its class name as Employee, with 4 attributes namely Emp_ID of data type integer, Emp_Name of data type String, Emp_Address of data type String and Emp_Salary of data type double. The plug-in generates related entity class of the designed class diagram. RTET is launched to generate a deployable system from the Employee entity class. The EJB Generator option is selected to generate the EJB class where PostgreSQL is chosen for the database. The Employee entity class is browsed for and is processed by RTET which identities and displays the attributes on its GUI as shown in Fig. 2. CRUD and viewAll functions to be generated in the EJB are selected by default.

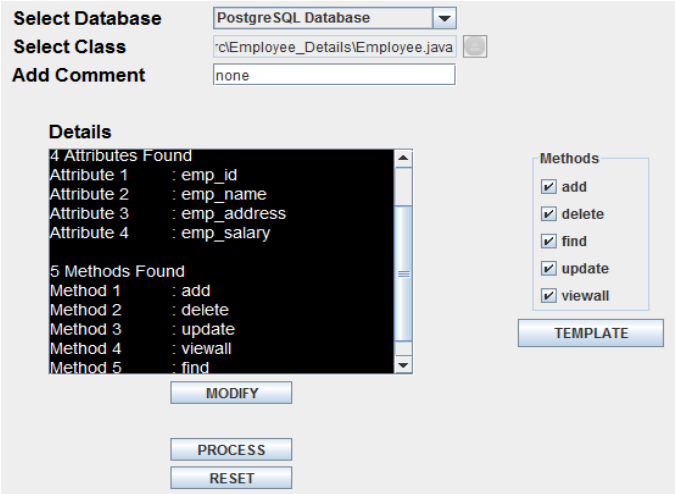

Figure 2. EJB Generator

The names, data types and number of attributes of the entity can also be modified. The template option can be selected to browse for existing templates or to create a new one in the available text editor. The EJB Generator component successfully generates the EJB session bean based on the preferences of the user as partly shown in Fig. 3.

```
// Method add
public int add(int emp_id, String emp_name, String emp_address, double emp_salary)
        throws SQLException, ClassNotFoundException {

    // Creating the attribute con for the Connection
    Connection con = null;

    // Creating the attribute s for the Statement
    PreparedStatement pstmt = null;

    // Creating the connection with the drive
    Class.forName("org.postgresql.Driver");
    // Creating the driver manager by using the URL, LOGIN and PASSWORD
    con = DriverManager.
            getConnection("jdbc:postgresql://localhost:5432/postgres","postgres","password");

    // The add query
    String addData = "INSERT INTO Employee VALUES(?,?,?,?)";
    // Add the Data
    pstmt = con.prepareStatement(addData);

    // Assigning the value
    pstmt.setInt(1, emp_id);
    pstmt.setString(2, emp_name);
    pstmt.setString(3, emp_address);
    pstmt.setDouble(4, emp_salary);

    // Add the data and assigning the value 1 if data has been added
    int add = pstmt.executeUpdate();

    // Close the s Statement
    pstmt.close();
    // Closing the con connection
    con.close();

    return add;
}
```

Figure 3. Generated EJB class codes

Based on the EJB session bean generated, RTET generates interfaces using Java Server Pages (JSP) or Java Server Faces (JSF) framework. 10 pages are generated for normal JSP, as shown in Fig. 4, where one of them is a JavaScript consisting of validation codes for the interfaces. The other nine pages are JSP pages that provide interfaces to interact with the business logic or CRUD functions and the database.

11 pages are generated for the JSF framework, quite similar to the normal JSP (Fig. 4). One among the pages generated, is a Controller.java class, which is used to interact between the interface and the business logic. 2 XML files Faces_config.xml and Web.xml are updated. The other pages are .jsp pages consisting of interfaces to interact with the business logic layer and database layer.

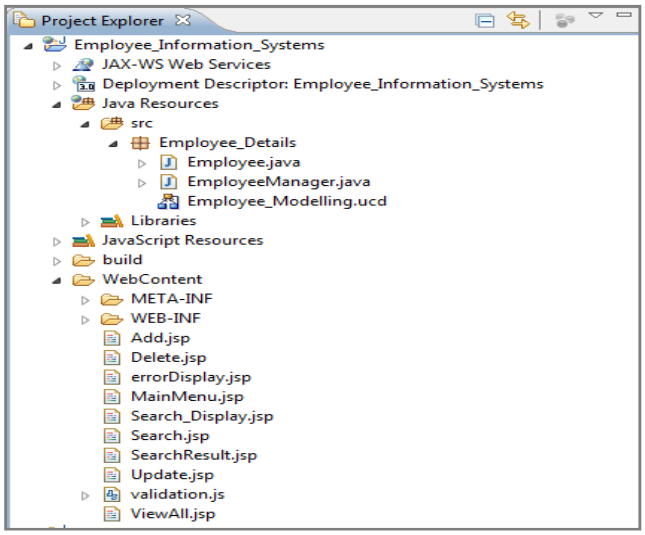

Figure 4. Generated JSP pages

Among the pages generated for both frameworks, a main menu page, as in Fig. 5, is generated which links all the other pages.

The interfaces generated allow the user to perform the following tasks:

- Add details in database table, as in Fig. 6
- Delete records from database table on primary key
- Update the details from the database table
- View all details from the database table
- Search for details using primary key

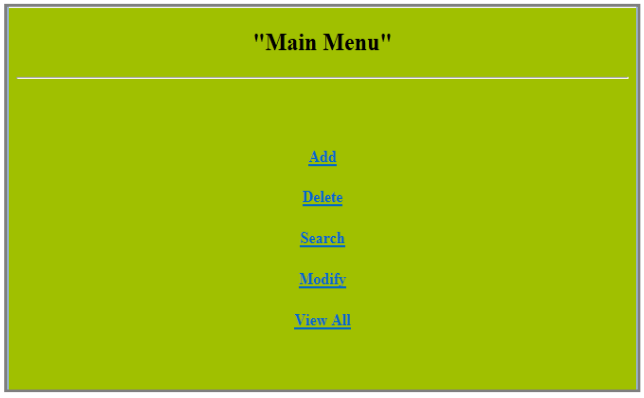

Figure 5. Generated main menu page with chosen background colour

Figure 6. Generated Add.jsp page

RTET also performs reverse engineering of any interface in JSP as shown in Fig. 7 (an add.jsp page). It processes the interface and identifies the attributes of the entity. Based on the selection of the user, the EJB session bean and the corresponding entity bean are successfully generated.

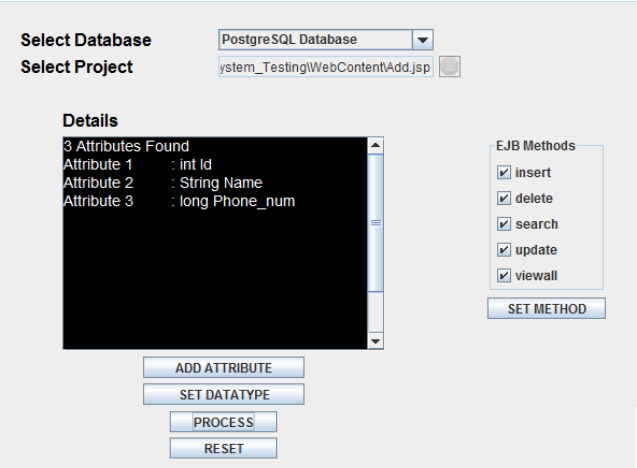

Figure 7. Reverse Engineering from an Add.jsp page

V. DISCUSIONS & EVALUATIONS

There are many features in RTET that makes a remarkable difference among existing round trip engineering tools. RTET seems to be the only round trip engineering tool that automatically generates a tiered EJB system with JSPs implementing the user interface, EJB session and entity beans for the business logic functions, and the data definitions for the tables in a specified database. MVC web applications can also be generated with JSF views and appropriate managed beans. RTET can also reverse engineer a JSP or JSF user interface into a complete EJB session bean and its entity bean. The code generation feature of most round trip engineering tools provide for only skeleton codes that simply implements UML functionalities. However, RTET generates codes beyond that and generates enterprise beans and interfaces to directly interact with the business logics. Further, RTET provides more flexibility to users by allowing choosing databases at client side, frameworks of interfaces and many more as

described earlier. RTET intelligently generates codes according to user defined templates and hence can be tailored to meet the specifications of new frameworks. With the template module, there are virtually no restrictions in the style and standard of codes that can be generated by RTET.

RTET has been compared with 2 advanced tools identified by E. Palacios-Gonzalez et al [5] - Borland Together and AndroMDA, on whether or not they have the ability to:

- Generate EJB Classes implementing CRUD functions
- Generate Interface in normal JSP pages
- Generate JSF pages and managed bean
- Reverse engineer interface into EJB
- Reverse engineer interface into entity bean
- Generate data definitions for database

Table IV shows a comparison of the three tools based on the above criteria, where ✓ means that the feature is supported by the tool and ✗ means that it is not supported.

TABLE IV. COMPARISON TABLE

Criteria	Borland Together [6]	AndroMDA [15]	RTET
Generate EJB Classes implementing CRUD functions	✓	✓	✓
Generate Interface in normal JSP pages	✗	✓	✓
Generate JSF pages and managed bean	✗	✓	✓
Reverse engineer interface into EJB	✗	✗	✓
Reverse engineer interface into entity bean	✗	✗	✓
Generate data definitions for database	✗	✓	✓

Although RTET is a powerful round trip engineering tool, it does have some limitations in terms of securing transactions, code generation for relationships between entities and lack of support for multiple languages. Also, automatic synchronization of the web pages, the EJBs and the model is currently missing.

VI. CONCLUSIONS & FUTURE WORKS

The use of Java EE architecture, libraries and frameworks of Java in RTET development makes it easier and more useful to perform round trip engineering tasks. RTET proves to be a reliable, real time code generator and reverse engineering tool. The tool automatically generates a tiered EJB system with JSPs implementing the user interface, session and entity beans for the business logic functions, and the data definitions for the tables in a specified database. MVC web applications can also be generated with JSF views and appropriate managed beans. RTET can also reverse engineer a JSP or JSF user interface into an EJB session bean and its entity bean. Unlike other round trip engineering tools, users of RTET can generate a deployable web application using only a few mouse clicks.

However, the development of RTET does not stop with these features. There are a number of enhancements that can be done to the system to address the current limitations and make it more powerful and useful for the developers' community. RTET currently generates only CRUD functions for entities. Future versions of RTET will support relationships between different entities and generate EJBs and user interfaces for these relationships. Some works can be done on securing the web applications using security constraints and securing the enterprise JavaBeans using declarative or programmatic security. Automatic synchronization of user interfaces, EJB session and entity beans, database tables and the models is also in our wish list. The current version of RTET tool supports only code generation in the Java Programming language and providing for only two Java frameworks, namely JSP and JSF. Thus, as future works, RTET primarily intends to support more programming languages, such as C++, C# and frameworks, such as Hibernate, Spring and Struts. RTET also aims at becoming a web based tool rather than a desktop application so as to enable better collaboration among developers from different places. Additionally, RTET is intended to be transformed into a plug-in for existing IDEs such as Eclipse and NetBeans and might be considered for being released as an open source software or project.

REFERENCES

[1] J. Mukerji and J. Miller, *"MDA Guide V1.0.1"*, Technical report, OMG, 2003, Accessed on 10th March 2012.

[2] A. Steelman and J. Murach, *Murach's Java Servlets and JSP*, 2nd ed., Mike Murach & Associates, 2008.

[3] F. Nimphius, *JavaServer Faces 2.0 Overview and Adoption Roadmap in Oracle ADF Faces and Oracle JDeveloper 11g*, Oracle White Paper, 2010.

[4] E. Palacios-González, H. Fernández-Fernández, V. García-Díaz, B.C.P. García-Bustelo, and J.M.C. Lovelle, *"A review of Intelligent Software Development Tools"*, in Proc. IC-AI, 2008, pp.585-590.

[5] E. Palacios-Gonzalez, H. Fernandez-Fernandez, V. Garcia-Diaz, B. C. P. GBustelo, J. M. Cueva Lovelle, and O. S. Martinez, *"General purpose MDE tools"*, International Journal of Interactive Multimedia and Artificial Intelligence, 2008, Vol. 1, Nº 1, ISSN 1989-1660

[6] Borland Software Solutions, "*Borland Together 2008"*, 2009.

[7] Rational Software, "*IBM Rational Rose – Data Sheet*", 2006.

[8] K. Odutola, A. Oguntimehin, L. Tolke and M. V. D. Wulp "*ArgoUML Quick Guide"*, 2001.

[9] T. Allegrini, "*Code generation starting from statecharts specified in UML*" Universita Degli Studi Di Pisa, 2002

[10] M. Boger, E. Graß and M. Köster, "*Poseidon for UML Users Guide*", 2007.

[11] M. Yuan, J. Orshalick, and T. Heute, *Seam Framework: Experience the Evolution of Java EE*, 2nd ed., Prentice Hall, 2009.

[12] J. Orshalick and N. Assar, "*JBoss Seam: Agile Ria Development Framework"*, Red Hat Inc., 2010.

[13] C. Walls, *Spring in Action*, 3rd ed., Manning Publications, 2011.

[14] D. Winterfeldt, *"Spring by Example"*, Version 1.2.1, 2012.

[15] AndroMDA Homepage , Available at http://www.andromda.org, accessed on 15th August 2012.

[16] A. Goncalves, *Beginning Java EE 6 Platform with GlassFish 3*, 2nd ed., Apress, 2010.

[17] D Steinberg, F. Budinsky, M. Paternostro and M. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed., Addison-Wesley Professional, 2008.

[18] Eclipse Modeling Framework Project (EMF) Homepage, Available at http://www.eclipse.org/modeling/emf, Accessed on 18 August 2012.