Information Systems Frontiers 5:2, 149–160, 2003

The Migration of Multi-tier E-commerce Applications to an Enterprise Java Environment

Terence C. Lau
Centre for Advanced Studies, IBM Canada Laboratory
E-mail: Lautc@ca.ibm.com

Jianguo Lu
Department of Computer Science, University of Windsor
E-mail: jlu@cs.uwindsor.ca

John Mylopoulos
Department of Computer Science, University of Toronto
E-mail: jm@cs.toronto.edu

Kostas Kontogiannis
Department of Electrical and Computer Engineering, University of Waterloo
E-mail: kostas@swen.uwaterloo.ca

Abstract. *As technology evolves, many organizations face the problem of migrating legacy applications from one technology base to another. We report on a case study involving the migration of legacy code into the IBM® WebSphere® Commerce Suite product. Specifically, we focus on the problem of migrating applications that use traditional database access techniques to applications using the Enterprise JavaBean (EJB) programming model. Our results include a practical methodology that facilitates such migration, as well as a tool that supports this methodology. The tool has been released on IBM's alphaWorks site.*

Key Words. *e-commerce, migration, database reengineering, Enterprise JavaBean, SQL, Net.Data, JSP, relational-object mapping*

1. Introduction

Multi-tier applications typically consist of a middle tier of web servers, web browsers at the front end, and database systems in the back end. Such applications have existed for years and many of them need to be migrated to new technology platforms. This paper describes our experience in migrating the IBM e-commerce framework, a multi-tier application, to a new framework that is based on Enterprise JavaBean technology. In particular, we focus on the migration of the presentation tier of the system using the relationships established in other tiers of the systems. Our results include a practical methodology to support client migration. In addition, we describe our experience in building a Net.Data-to-JSP (Java Server Page) Helper Tool that is currently being used by developers in migrating e-commerce websites developed using IBM WebSphere Commerce Suite, from earlier versions to more recent ones. Unlike most research on software migration that focuses on monolithic systems, our work addresses migration problems where both the source and target systems involve multi-tier architectures that use different programming languages.

The rest of the paper is organized as follows. We first introduce in Section 2 background information concerning the source and target platforms for our migration task. Section 3 describes the migration process for the IBM Websphere Commerce Suite product. In Section 4 we present an implemented migration tool, while Section 5 introduces a generic migration model from any legacy system to an EJB-based architecture. Section 6 compares our work with related research, and Section 7 offers conclusions and directions for further research.

2. Background

2.1. Enterprise JavaBean

EJB (Enterprise JavaBean) is a distributed component framework that provides transaction, security and persistence services in a distributed multi-tier environment (IBM, 2001; iPlanet Application Server Migration Guide, 2000). The EJB framework separates business logic from low-level details, so that developers can concentrate on a business solution. With the growing popularity of the EJB framework, more and more legacy systems (web-based or otherwise) are being transformed into an EJB architecture. Typically, a legacy system includes a large number of SQL queries. On the other hand, in an EJB-based architecture those SQL queries are not used directly. Instead, the popular model-view-control design pattern is adopted, where queries are wrapped inside EJBs so data are accessed through EJBs instead of SQL statements. A typical EJB-based architecture is shown in Fig. 1.

At the center of the architecture is the EJB container that manages a set of enterprise beans. The beans access backend systems, typically relational databases. The web container typically uses JSP to access the EJB, transforms the JSP to HTML, and serves the HTML to the browser. EJB client applications other than JSP, such as Java applets or other systems, can also access the enterprise beans.

The enterprise beans in the middle tier function as wrappers over various systems, most notably relational database systems. There are two kinds of enterprise beans: *entity* and *session* beans. Roughly speaking, session beans represent activities, while entity beans represent entities of the application. An entity bean is a persistent object stored in a storage system such as a database system. In a simple scenario, one bean corresponds to a row in a table. The selection of certain rows of the table corresponds to the selection of a group of beans satisfying a given condition. Beans are selected by using a finder method that contains an embedded SQL statement. We are interested in entity beans.

By using EJBs, web designers no longer need to learn the details of database structure. In addition, changes in the backend database are shielded by the beans, making the maintenance of Java Server Pages easier.

2.2. Java Server Pages

Java Server Pages (JSPs) enable Enterprise Java applications to create dynamic content for browser-based clients. JSPs constitute a presentation-centric method for developing *servlets*, i.e., Java programs that are executed by a web server and have their output directed to a client browser. JSPs support a reusable component model based on JavaBean components and custom actions.

Using JavaBean components web page designers can focus on presentation while application developers can develop specific components to process and return data to be used in a page. JavaBean components can also be used elsewhere in the application, as they are reusable and portable.

2.3. Net.Commerce and WebSphere Commerce Suite

IBM WebSphere Commerce Suite and its earlier versions, Net.Commerce (Lau et al., http://alphaworks.ibm.com/tech/netdatatojsp) are platforms for building

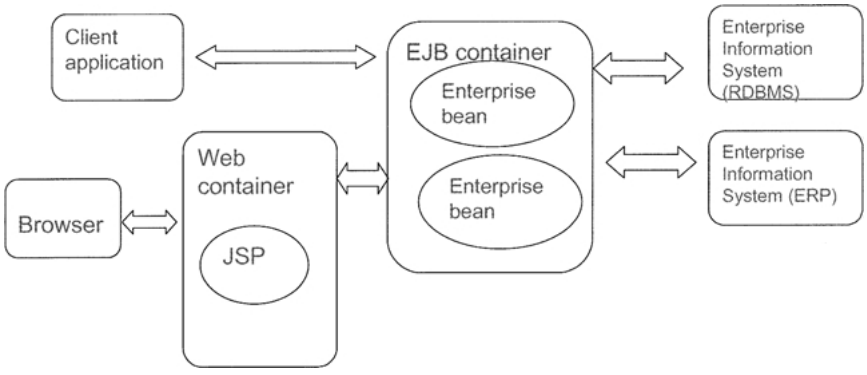

Fig. 1. EJB architecture.

e-commerce applications. WebSphere Commerce Suite supports functionalities ranging from product catalogue browsing and payment processing, to product promotion and auction. Starting in 1995, Net.Commerce went through five major revisions, was renamed WebSphere Commerce Suite, and more recently WebSphere Commerce. In November 2000, the WebSphere Commerce Suite offered a number of enhancements over Net.Commerce, affecting its database schema as well as its programming model. The enhancements emphasize clear programming structure and conformance with the EJB model. Since that time, existing WCS customers who want to transition to newer versions of WebSphere Commerce Suite need the help of tools as well as methodological guidance to migrate their legacy e-commerce applications to the target environment.

2.3.1. The IBM Net.Commerce and WebSphere Commerce Suite Version 4 programming model. The early versions of Net.Commerce and WebSphere Commerce Suite are based on a programming model built on C++ and use commands, tasks and overridable functions. Commands are C++ components servicing major functions, such as placing an order. Tasks represent sub-units of work within a command, such as checking inventory or calculating the price for an order. Overridable functions are the C++ components that actually perform the work of the tasks. Customers usually introduce their own business logic by replacing the overridable functions provided by WebSphere Commerce Suite with their own. When results and responses are presented to users, Net.Data® macros are invoked to access the backend database, retrieve the appropriate information, and then present it on Web pages.

2.3.2. The IBM WebSphere Commerce Suite Version 5.1 programming model. Table 1 highlights the differences between the two e-commerce systems. For the presentation language, the old system uses Net.Data, while the new one uses Java Server Pages (JSP), a widely accepted standard for dynamic web pages. For the business logic, the old system uses SQL, in many cases directly embedded in C++ programs and Net.Data scripts, while the new system uses EJB. For the programming language, C++ is used in the old system, and Java in the new. As for the programming model, the old system intertwined many things in the same place and does not have a clear separation between the view, model, and control of the system. The new system adopts the popular Model-View-Control (MVC) design pattern by using EJB.

Table 1. *WebSphere Commerce Suite (WCS) programming models: Now and before*

	Before (WCS V4.1 and prior)	Now (WCS V5.1)
Presentation language	Net.Data	JSP
Business logic	SQL	EJB
Programming language	C++	Java
Programming model	No clear MVC	MVC

3. Migration Process

3.1. The migration tasks

The transition of Net.Commerce and earlier versions of WebSphere Commerce Suite to WebSphere Commerce Suite Version 5.1 consists of three major processes as illustrated in Fig. 2.

- *Migrating software stacks*: The software stacks include the system and application software packaged with WebSphere Commerce Suite Version 5.1 that provide basic system functions and third-party application functions. They also include the operating system, the database system, the web server, the security system and the like.
- *Migrating WebSphere Commerce Suite infrastructure*: The WebSphere Commerce Suite infrastructure consists of the basic components shipped with the product. Customers build their specific solutions on this infrastructure. The infrastructure includes tools for building a Web site based on WebSphere Commerce Suite, runtime infrastructure, the WebSphere Commerce Suite database schema, the WebSphere Commerce Suite class library and object entities.
- *Migrating customer assets*: Customer assets include those accumulated, populated, customized, or extended by customers in their commerce sites. These assets may be database data, pages designed specifically for the Web site, database schema extensions, and new or customized business logic in commands and overridable functions.

The rest of the paper focuses on the migration of customer assets. In particular, the transition to WebSphere Commerce Suite Version 5.1 involves the conversion of the following major assets:

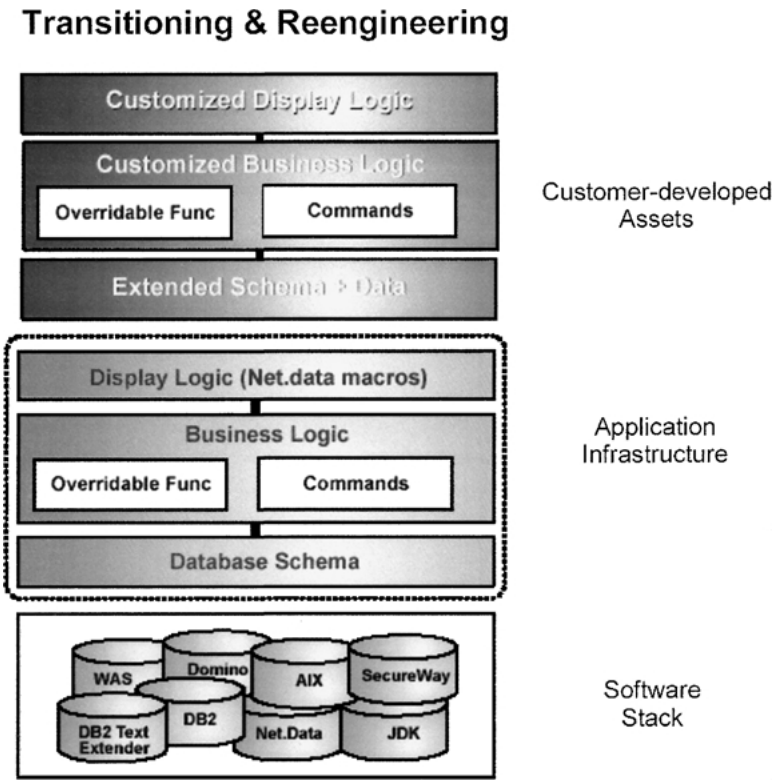

Fig. 2. *Migration processes.*

- Data is migrated to the WebSphere Commerce Suite Version 5.1 database format and schema, and accessed using the new object models of EJB and data beans.
- Net.Data macros are replaced by JSP. The SQL in Net.Data macros are replaced with JSP calls to WebSphere Commerce Suite Version 5.1 data or business EJB. Business logic within a Net.Data macro should be moved into a command to isolate the view from the model.
- Business logic is converted from C++ to Java commands. Overridable functions are replaced by EJB task commands. Major C++ commands are replaced by EJB controller commands.

In the following we discuss the migration of Net.Data macros to JSP.

3.2. The migration process

Our process consists of two main phases: i.e., reverse engineering and forward engineering.

In the reverse engineering process the following mappings are established:

(1) *Schema-to-Schema Mapping (SSM)*: Relational database schema, mapping data from the source to the target system.
(2) *Command to Command Mapping (CCM)*: We define a command as a component of the source system that accomplishes a particular task and has a comparable counterpart in the target system, such as `addUser` command.
(3) *Entity to Relation Mapping (ERM)*: The relationship between the target system EJB Object (entity beans) and database relation information.
(4) *Component to Entity Mapping (CEM)*: This is the mapping between the target system JavaBean components and Enterprise bean relations.

Taking the result of reverse engineering as input, the forward engineering process proceeds as shown in Fig. 3.

Fig. 4 depicts in detail the various data sources and processes involved in our migration system. We now explain the purpose and relationships between various elements of the system.

Input: Net.Data, CCM, SSM, ERM, CEM.
Output: Command, Component JavaBean
Steps:

1. User select a Net.Data file and the SQL inside Net.Data
2. Translate the selected SQL into the SQL in terms of the new schema using SSM.
3. Compare the translated SQL with the SQLs inside entity beans, and identify the relevant entity beans using ERM.
4. From the entity beans identified the step 3, find EJB components using CEM.
5. Using this information in step 4, the set of JavaBeans can be reduced if the user can identify which JavaBeans are used by these commands in the target system.
6. If there are commands that run before or after the SQL execution in the source system, find the equivalents of these commands in the target system using CCM.

Fig. 3. *Forward engineering process.*

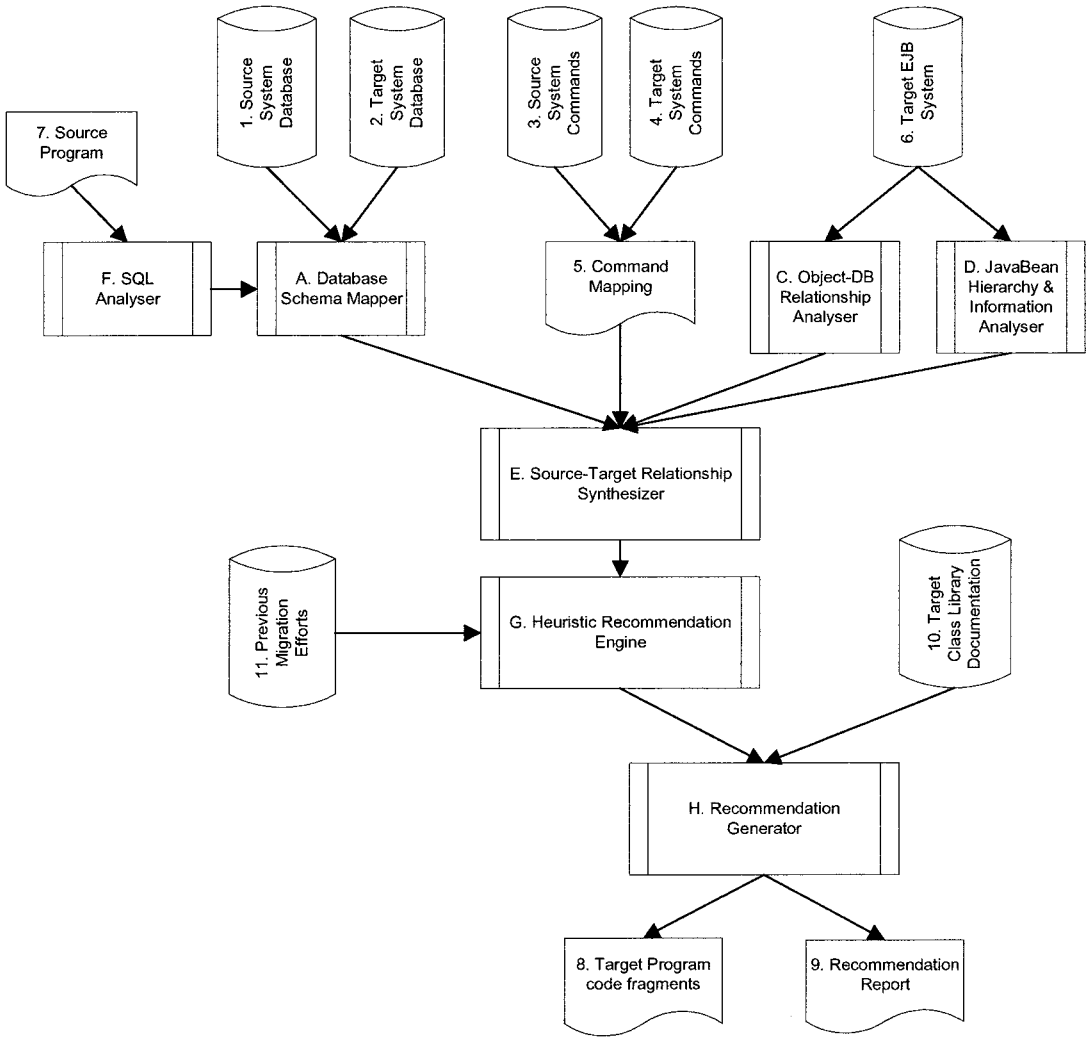

Fig. 4. *System diagram.*

Prior to the first use of the tool the target application source code is analysed. The Object-DB analyser (Item C) extracts information from the target application to determine relations between Entity Beans and the underlying database. This is accomplished by parsing WCS source code from the target application and needs only to be run once provided the underlying source remains static. In general, there is an Entity Bean for each table in the database that provides read/write access to data elements and performs selection operations on the data. We call this relationship the Primary relationship between the table and the associated Entity Bean. A secondary relationship exists if the Entity Bean uses data from other tables in any selection methods. All of the finder methods of the entity beans are also recorded.

We then examine the client JavaBean Components to determine their usage of entity beans (Item D) and any containment relationships that exist between the JavaBean Components. This process is also run once and needs only to be re-run when the underlying JavaBean code changes.

During initialisation of the migration tool all relevant migration information is read into the system. The Database Schema Mapper (Item A) process accepts as input the source-to-target database schema mapping relationship as specified in a configuration file. This relationship states what target database table/column(s) map to what target database/column. The process generates an internal structure representing this mapping relationship. The relationship between the source database tables and target database tables can be 1-to-1, 1-to-many, or many-to-1.

The Source-to-Target Relationship Synthesizer (Item E) integrates migration information so that given a table from the source application, it is possible to determine all possible JavaBean components from the target application that could have access to similar data in the target system.

After initialisation, the tool accepts as input a source file and extracts all the SQL statements contained in the file. The SQL Analyser (Item F) extracts the column(s) and tables used in a particular SQL statement. The following example illustrates the recommendation process.

Given the following SQL statement:

```
SELECT A, B, G, H
FROM X, Y
WHERE X.A = Y.G
```

The SQL Translator translates the query into

```
SELECT A', B', G', H'
FROM X', Y', Z'
WHERE X'.A' = Y'.G'
```

Suppose that from the database schema mapping *SSM* we have mappings such as:

```
(source table) X -> X' (target tables)
(source table) Y -> Y', Z' (target tables)
```

Now we compare the translated query with the queries inside entity beans. Using *ERM* as below

```
X', Y', Z' (Target system table)
-> X_EntityBean,
   Y_EntityBean,
   Z_EntityBean,
```

we can deduce that `X_EntityBean`, `Y_EntityBean`, and `Z_EntityBean` are relevant.

We then determine the set of JavaBean components that use these entity beans based on the following *CEM*:

```
X_EntityBean, Y_EntityBean, Z_EntityBean
-> A_JavaBean,
   B_JavaBean
```

The number of entity beans that a particular JavaBean component uses is treated as criterion to rank the recommendations. The logic behind this ranking is that the more data coverage a component has, the better the chances that it will have a combination of methods that will return data similar to the data retrieved from the SQL statement.

The set of ranked JavaBean components is passed on to the Recommendation Generator (Item H) that produces the output to the user in the form of a report or to a GUI.

4. Net.Data-to-JSP Migration Helper Tool

We have developed a tool to aid the migration of Net.Data macro files used in client e-commerce projects to JSPs. The tool was developed following the architecture and processing model outlined in the previous section.

The process of converting a commerce site using Net.Data macros to a site using JSP templates is by no means an automated task. There are many design decisions and code conversions to be carried out by the

developers responsible for the migration (particularly when developing business logic components). Thus, the Net.Data Migration Helper Tool is a reference tool that helps the development team plan their Net.Data migration and gives them guidance on where to begin. The tool can be used at two points in the migration process:

(1) When the migration planners are trying to determine how much work needs to be done on each file in order to create a project schedule for migration. The tool gives a high level view of how many functions each Net.Data file has, and how many standard and customised tables are involved.
(2) When the developer is working on creating a particular JSP template to replace a Net.Data file. The tool acts as a reference for the mapping between Net.Commerce Version 4 and WebSphere Commerce Suite Version 5 tables, for the functions in the file and recommended beans that should be used to perform the same function in a JSP.

4.1. Net.Data scripting language

IBM's Net.Data product enables developers to create dynamic web pages using data from relational databases and other back-end systems. Net.Data is a script language that enables a developer to specify the layout of Web pages, calls functions that are defined by macro, and defines variables and functions.

Net.Data macros contain two parts: the declaration part and the presentation part. Fig. 5 illustrates the structure of a Net.Data macro.

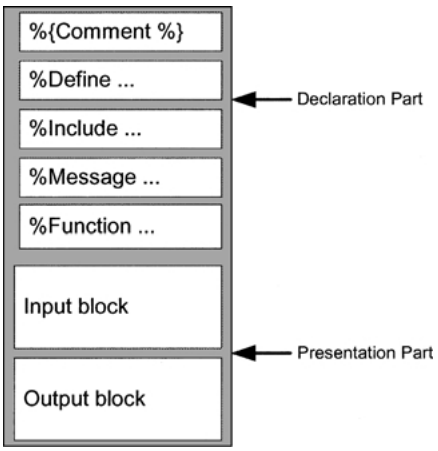

Fig. 5. *Net.Data language.*

The function block is what we are more interested in. A common function block has the following layout:

```
%function (dtw_odbc) name () {
   SELECT distinct safname, samname,
salname, shrfnbr, satitle
   FROM shopper, shaddr
   WHERE sashnbr = shrfnbr and
shlogid = '$ (SESSION_ID)' and
sanick = '$(SESSION_ID)'
   %REPORT{
     %ROW{
       <center>
         HTML formatting information for
rows returned from the SELECT statement
       </center>
     %}
   %}
%}
```

The SQL statements that we analyse are generally found in the function blocks of a macro.

4.2. Objective of the tool and rationale

Because of the significant differences between Net.Data and JSP, and the fact that customers might want to make functional changes during the migration, the objective of the tool is not to perform complete automatic conversion of one Net.Data macro to one JSP. Rather, it is a helper tool. Although the conversion process requires human involvement, the tool significantly reduces the total effort required of developers. A good list of recommendation can save a significant amount of effort and time during the first part of the conversion process.

The rationale for this approach is two-fold:

First, from an implementation feasibility point of view, automatically translating from Net.Data to Java is extremely difficult. Net.Data macros and JSPs are very different in form and implementation. It is not easy to predict the mapping from an SQL statement in a Net.Data macro to a particular data bean. Allowing the users to participate in the mapping makes the tool much more useful.

Secondly, because of the many enhancements in WebSphere Commerce Suite Version 5 that customers want to take advantage of, they will very likely prefer not to perform an automated translation of the Net.Data. The database schema includes extensive enhancements from Net.Commerce Version 4 to

WebSphere Commerce Suite Version 5, as well as improved function and design of some major components. It is neither practical nor useful to capture and fix this information in one shot, because customers are likely to customize the product and need to modify the information themselves. A more practical approach is to let the helper tool provide a more general recommendation initially, and allow users to add their own wisdom to build up the tool's knowledge.

This approach also paves the way for a potentially powerful function, namely a learning capability. As customers use the tool and indicate their choices, the information is captured so that more specific recommendations can be made in future uses. Customers usually have collections of similar Net.Data macros. After a lead developer trains the tool for one macro, it can then give out very useful recommendations for the rest of the collection to other developers. Similarly, this tool can be used to capture the experience of one customer situation so that it can be re-used to deal with similar situations. The learning capability is not included in the first release of the tool.

4.3. Tool features

The features of the tool are of two categories. (a) In the reverse engineering process it extricates the relationships between the data beans, access beans, and entity beans. Also, it establishes the connection between the database and the enterprise beans. (b) In the forward engineering process, it performs SQL translation and recommends to the user Data beans are the JavaBean components (data beans) to use. In order to fully understand how to use the particular data bean and to confirm that it is the right object for the task the tool provides a link to the JavaDoc documentation of the data bean.

4.3.1. Reverse engineering features

Schema mapper. This feature helps the user to capture custom database schema mapping between the Net.Commerce Version 4 and WebSphere Commerce Suite Version 5 systems. Users first prepare an XML file specifying the database relationship between the two versions of WebSphere Commerce Suite. This XML file is placed in the configuration directory of the tool and used later on for database and bean analysis, mapping, and recommendation.

Source code crawler. The source code crawler collects the data bean and access bean information from the EJB source code in the system and generates an XML file containing such information for subsequent recommendation.

SQL collector and extractor. The SQL collector parses Net.Data macro files provided by the user, extracts the SQL statements in an interactive manner, and generates an XML file containing the SQL information for subsequent analysis and processing. The SQL extractor has a similar function to the SQL collector, but works in a batch mode.

4.3.2. Forward engineering features

Bean recommendation. Fig. 6 is a screen-shot of the base recommendation tool.

The left pane is a full editor into which the Net.Data macro loads. The tool enables the user to cycle through each of the SQL statements found in the macro. The right pane shows the table and column mapping information and the recommended set of data beans of the active SQL statement.

The tool can produce a report for the loaded Net.Data macro that lists all the SQL statements in the macro, the function it came from, and the recommended data beans and their related access beans. When the database schema of WebSphere Commerce Suite is modified by the customer, the tool can also deal with the customised tables used in the SQL statements.

As an example, when a Net.Data page is loaded into the tool, the macro is parsed to extract all the SQL statements. After that, the SQL statement is parsed and relevant information is extracted, such as columns and tables that are used. A typical sequence of activities by a user is as follows, with reference to panes [A] to [F] in Fig. 6:

1. Select the SQL statement in the Net.Data macro as depicted in the top-left pane [A] of Fig. 6.
2. Note the Version 4 to Version 5 table correspondence in pane [B] of same figure.
3. Note the Version 4 to Version 5 column correspondence in pane [C].
4. Open the recommendation pane and note the related data beans in pane [D].
5. Go through the list selecting data beans of interest in [D], and note the detailed information of related data beans in pane [E].
6. Using the information in pane [F], compare against getter methods in pane [E] to narrow down the data bean selection.

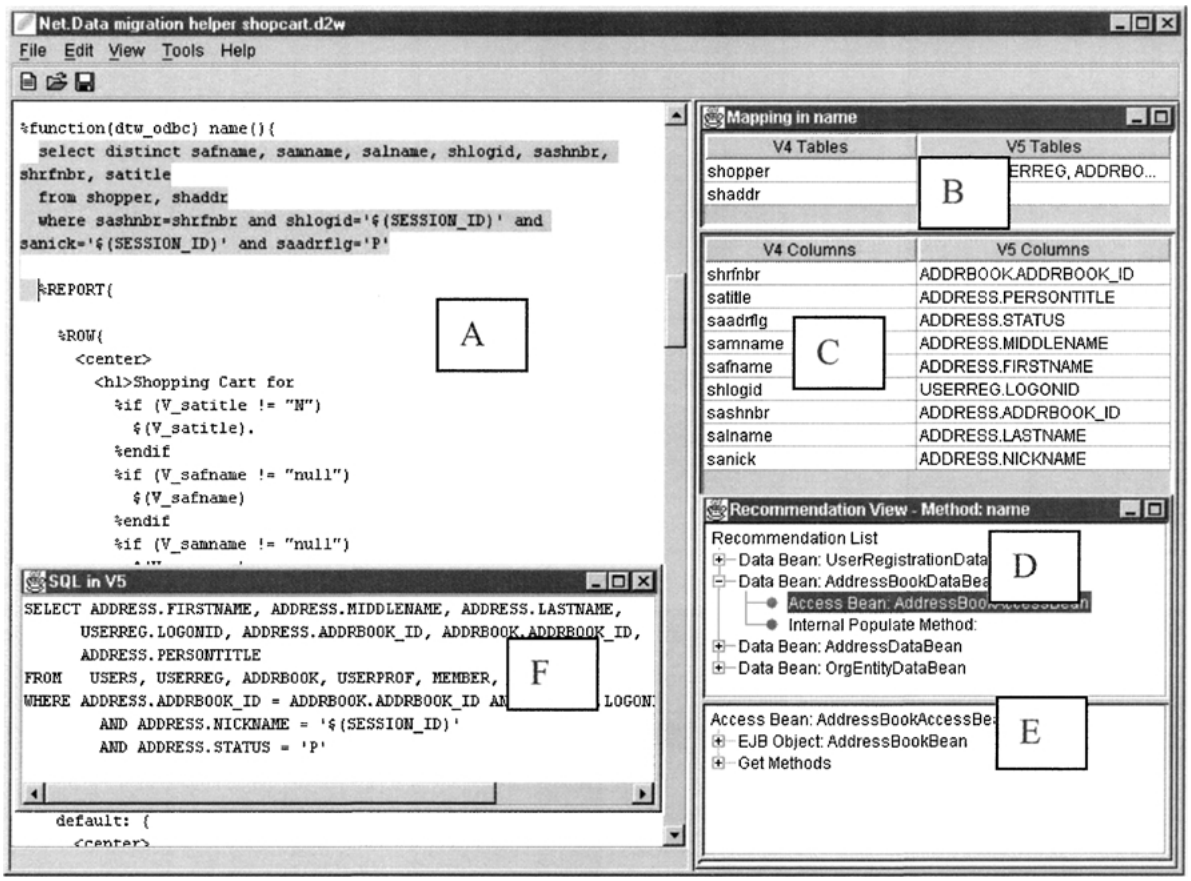

Fig. 6. *Tool window view.*

After going through the above steps, the possible output JSP code corresponding to the `name()` function in Net.Data file in the database schema mapper, as deduced from the JavaBeans Hierarchy and information analyser and the source-target relationship synthesizer, might be:

```
......
<jsp:useBean id = "addressbook"
class = "com.ibm.commerce.user.beans.Address
BookDataBean" scope = "page"/>
<%com.ibm.commerce.beans.DataBeanManager.
  activate (addressbook, request); %>
</jsp:useBean>
<!-- HTML content -->
......
<center>
<h1> Shopping Cart for
<% = addressbook.getTitle() %>
<% = addressbook.getFirstName() %>
<% = addressbook.getMiddleName() %>
<% = addressbook.getLastName() %></h1>
......
```

SQL translator. The SQL translator parses SQL statements from a Net.Commerce Version 4 system, maps out the corresponding WebSphere Commerce Suite Version 5 database information based on the Version 4 to Version 5 database mapping information in the tool, and constructs corresponding WebSphere Commerce Suite Version 5 SQL statements.

4.4. Tool deployment

This tool was first released in June 2001 on IBM's alphaWorks® Web site (Lau et al., http://alphaworks.ibm.com/tech/netdatatojsp). The primary target audience at that time was members of the WebSphere Commerce Suite Service Teams embarking on migration projects, for project planning and code development.

The database schema mapping is from WebSphere Commerce Suite Version 4.1 to the new Version 5.1 schema.

5. A Generic EJB Migration Model

Based on our experiment for the migration of Websphere Commerce Suite, we identified a generic migration model and the research challenges in moving from the multi-tier applications to EJB-based architectures other than IBM products. When reengineering such applications, we will face two challenges that are depicted in Fig. 7:

- When the enterprise beans are already provided, how to translate the queries embedded in the legacy code to the equivalent EJB client code?
- When the enterprise beans are not provided, how can we produce the beans, especially the finders and the queries inside finders consistently with existing legacy queries?

We propose to use SQL-EJB mediator to solve the first problem, and use view selection to solve the second. In general these two tasks are not performed independently. The reengineering is an iterative process with intervention from EJB designers. In a typical scenario the EJB designer defines some entity beans first, then uses the SQL-EJB mediator to locate legacy SQL queries that cannot be reproduced using the enterprise beans. Those queries are then used as the basis for EJB generation. By using the EJB generator, enterprise beans and the finder methods are recommended for the designer to choose. The selected beans are added into the existing system and we can iterate this process, run the SQL-EJB mediator once again.

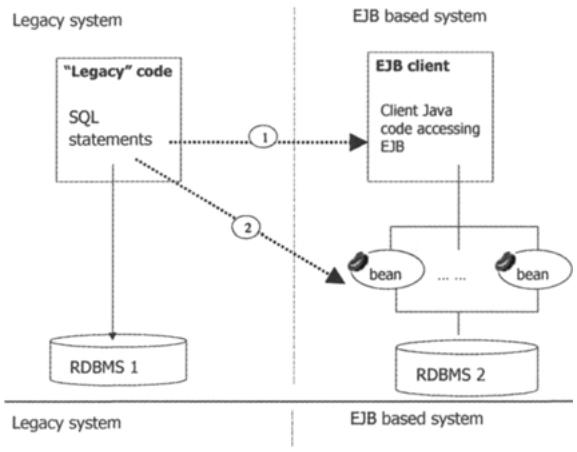

Fig. 7. EJB migration problems.

As for the EJB generation from a set of queries and their schema, there are two approaches. One is the black-box reengineering. In this approach, the SQL statements are not analysed. Instead, they are directly copied into the methods of enterprise beans. This kind of reengineering requires generating some scaffolding code for the method so that it can access the database. In general, this approach will use session beans.

On the other hand, the white-box reengineering approach is much more complicated. This will require the generation of entity beans and the rewriting of the queries into object interface. Several queries may be combined into one method in one entity bean, or one query may be split into several methods in different beans, or as illustrated in Fig. 8. This is a good long-term solution offering a clean object-oriented architecture.

Currently the EJB client code in JSP is not automatically generated. We are using query-rewriting techniques to translate the SQLs to fragments of EJB client code. Furthermore, in a more general situation when EJB architecture is not available yet, the EJB architecture and the finder methods as well as the SQLs inside the finder methos will be generated.

6. Related Work

There are several tools and methodologies for migrating EJB applications from one platform to another (iPlanet Application Server Migration Guide, 2000; Tech Metrix Research, 2000). These tools generally target migration tasks where the source system is already using EJB technologies and are therefore addressing a simpler problem than that addressed here. Moreover, our work takes into account specific requirements dictated by the WebSphere Commerce Suite.

In addition, there are tools for mapping database applications to EJB architecture (Takagiwa et al., 2001). However, these focus on the problem of mapping schemata to Enterprise JavaBeans, instead of migrating SQL code. Besides, the mappings supported by these tools are typically either straightforward or have to be performed manually. Such tools are most effective for the problem of object-relational mappings.

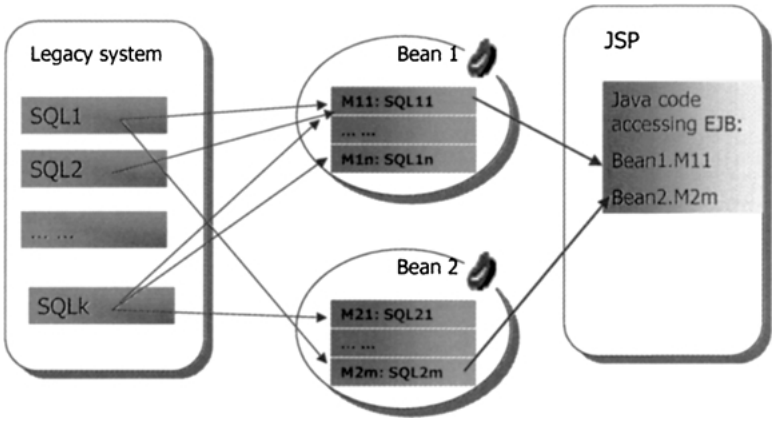

Fig. 8. *White-box reengineering.*

In database reverse engineering and schema mapping (Jahnke, Schafer, and Zundorf, 1996), a popular approach is to map a relational schema to an object schema directly. This paper addresses three additional aspects. First, we map the relational schema to another relational schema. Second, we extract and enrich the schema mapping from the hand-coded schema mapping data provided by IBM that are recoded in XSL format. Finally, we use the schema mapping to translate legacy SQL queries.

7. Conclusions and Future Work

We have presented a practical methodology fort the migration of multi-tier applications that access databases at the backend to ones using the EJB programming model. The methodology is supported by a tool developed for the IBM WebSphere Commerce Suite. The tool has been implemented and is being used by developers working on migration tasks. This work has also identified a number of areas that we believe are fruitful for further investigation:

Firstly, the recommendation ranking technique is rather simple in the current release of the tool and we believe that it can be improved. We are investigating how the following changes to the Recommendation Engine will improve the quality of the recommendations:

(1) Use the column usage information as a basis for doing data coverage analysis.
(2) Analyse the containment relationships that hold between data beans and the inheritance relationships between data and access beans, to see if these relationships relate to actual usage of the beans; if so, modify the ranking scheme to account for this.
(3) Retain user experience from previous migration efforts to aid developers encountering similar tasks in different migration projects.

Secondly, the tool can help programmers interactively produce the JSP code, and especially the EJB client code in JSP. We are currently working on the following tasks:

- The generation of EJB client code using query rewriting technology based on existing EJB framework (Lu, 2002);
- The generation of entity beans and session beans from the legacy database systems when they do not exist (Lu, 2002);
- The translation from Net.Data to JSP code, especially the HTML part.

Acknowledgments

We would like to thank the anonymous reviewers for their detailed comments. In developing the migration process and the Helper Tool, we had valuable help and input from Erik Hedges, Emily Xing, members of the E-Commerce Development Team and E-Commerce Engagement Team within the IBM Canada Laboratory.

The University of Toronto and Waterloo teams are grateful to the Canadian Consortium for Software Engineering Research (CSER), the IRIS Network of Centres of Excellence, and the Natural Sciences and Engineering Research Council of Canada for financial support of this research.

This paper represents the views of the authors rather than IBM.

References

Behm A, Geppert A, Dittrich KR. On the migration of relational schemas and data to object-oriented database systems. In: *Proceedings of the 5th International Conference on Re-Technologies for Information Systems*, Austria, 1997.

Bergamaschi S, Garuti, A, Sartori C, Venuta A. The object wrapper: An object oriented interface for relational databases. *Euromicro* 1997.

Brown K. *Handling N-ary relationships in VisualAge for Java*. www.ibm.com/vadd, Aug. 2000.

IBM, IBM Net.Data Reference. Version 7, http://www4.ibm.com/software/data/net.data/, June 2001.

IBM, IBM WebSphere Commerce Suite, Programmers Guide. Version 5.1 Second Edition. http://www4.ibm.com/software/webservers, March 2001.

In2J, Automated Tool for Migrating Oracle Pl/SQL into Java, www.in2j.com, Apr. 2001, Iplanet.

iPlanet Application Server Migration Guide. Version 6.0. http://docs.iplanet.com/docs/manuals/ias/60/migrate/http:///, May 2000.

Jahnke J, Schafer W, Zundorf A. A design environment for migrating relational to object oriented database systems. In: *Proceedings of the International Conference on Software Maintenance*, pp. 163–170. IEEE Computer Society Press, 1996.

Kassem N. and the Enterprise Team. *Designing Enterprise Application with the Java 2 Platform, Enterprise Edition*. Sun Microsystems, http://java.sun.com, Oct. 3, 2000.

Lau T, Lu J, Mylopoulos J, Hedges E, Kontogiannis K, Xing E, Crowley M. *Net.Data to JSP helper*. IBM alphaWorks, http://alphaworks.ibm.com/tech/netdatatojsp.

Lu J. Reengineering database applications to EJB based architecture. In: *CAiSE*02, 14th Conference on Advanced Information Systems Engineering*. Toronto, May 27–31, 2002.

Miller RJ, Haas LM, Hernández M. Schema mapping as query discovery. In: *Proceedings of the Twenty-Sixth International Conference on Very Large Data Bases (VLDB)*. Cairo, Egypt, Sept. 2000.

Moving from IBM WebSphere 3 to BEA WebLogic Server 5.1. TechMetrix Research, Sept. 2000.

Ramanathan C. Providing object-oriented access to existing relational databases, PhD dissertation, Mississippi State University, 1997.

Sun Microsystems, Enterprise JavaBeans 2.0 Specification., http://java.sun.com/products/ejb/2.0.html, 2001.

Vermeer MWW, Apers PMG. Reverse engineering of relational database applications. In: *Proceedings of the Fourteenth International Conference on Object-Oriented and Entity Relationship Modeling (ER'95)*, Dec. 1995.

Takagiwa O. et al. *Programming with VisualAge for Java Version 3.5*. IBM RedBooks, www.ibm.com/redbooks, April 2001.

Yan L, Miller RJ, Haas LM, Fagin R. Data-driven understanding and refinement of schema mappings. *SIGMOD* May 2001.

Dr. Terence C. Lau is a senior research associate at the Centre for Advanced Studies, IBM Toronto Laboratory, IBM Canada, and an adjunct associate professor at the Department of Electrical and Computer Engineering, University of Waterloo.

Dr. Jianguo Lu is an associate professor at the Department of Computer Science, University of Windsor.

Dr. John Mylopoulos is a professor at the Department of Computer Science, University of Toronto.

Dr. Kostas Kontogiannis is an associate professor at the Department of Electrical and Computer Engineering, University of Waterloo.