Enterprise Java Applications and SAP R/3 System Integration Using JCO

Jitao Yang[1], Hongqi Su[1], Yuanfeng Wu[1] and Junwei Liu[2]

[1] Department of Computer Science, China University of Mining & Technology, Beijing 100083, P.R. China iipapaya@gmail.com shq@cumtb.edu.cn yuanfengwu@126.com
[2] Industry Application Division, Tsinghua Tongfang Co. Ltd, Computer System Business Group, Beijing 100085, P.R.China liujunwei@thtfpc.com

Abstract. Enterprise computing often takes the form of automation islands, and lots of business operations must be built on these islands. Then reusing existing components and creating integrated, flexible, reliable applications which can be combined for maximum productivity have become the imperative for an enterprise to survive and thrive in the new competitive and dynamic business environment. In the company with SAP as its ERP software, it is an obvious choice for the enterprise to target SAP R/3 Enterprise as its data source. But the question is how to achieve the integration when the company used Java EE as its platform for developing enterprise applications? JCO (Java Connector) is a new and economical solution that is introduced in this paper for the integration.

Keywords: *Data synchronization, Design pattern, Electronic data interchange, Business process integration, SAP, JCO*

1. INTRODUCTION

Business collaboration through enterprise application integration is an absolute competitive differentiator in an increasingly global economy, and the goal of enterprise integration is to provide timely and accurate exchange of consistent information between business functions to support strategic and tactical business goals in a manner that appears to be seamless [1]. However, problems often emerge from overly ambitious or imprecise requirements resulting from inadequate plans for integrating different systems.

Integration of information systems is expensive and time consuming. A lot of labor costs can be traced to the storage and reconciliation of data. In addition, most of codes in corporate software systems are dedicated to moving data from system to system. In this paper, we describe the challenges associated with the integration of SAP R/3 Enterprise and provide a road map toward a solution.

2. PRELIMINARIES

2.1 Problem Description

In the company using SAP R/3 as its ERP software, it is clear for the company to use SAP as its data source, and integrate the other non-SAP systems, such as a e-business system based on Java EE platform, a .NET platform and so on. The integration requires regular or real-time customers, materials, and other main data from the SAP ERP system, ensuring the consistency of the data used in the business operations and the internal SAP ERP main data.

Currently, SAP XI (Exchange Infrastructure) will complete this job well, but the cost is expensive and its configuration is relatively cumbersome. Then, how to design a flexible using and simple configuration data synchronization framework based on the enterprises' business operation reality, which not only let the ERP software play the central role in the allocation of enterprise resources, but also maintain the existing heterogeneous platform systems run smoothly and stably without a lot of changes, is more concerned by the ERP enterprise.

Based on an actual project, the paper provides a solution using JCO to make the other heterogeneous systems' developers and the end users use the SAP data in a comparatively transparent manner.

2.2 Data Classification

The data required by the enterprise's business operation can be broadly classified into the following two types:

The first category of data: basic data for business operation, such as the types of customers, the customers' respective sale regions, province information, products transportation methods and etc. Such data is characterized by relatively small changes, and belongs to system enumeration values.

The second category of data: main data for business operation, such as material main data, customer information main data, shipment information and BOM (Bill of Material) data. Features of such data will be incremental or periodically changed which means have to be dynamic updated, and its real-time requirements are relatively high.

These two types of data need to design two different data synchronization programs.

2.3 Data Synchronization Design

Based on the reality, we designed a middle layer between the SAP ERP system and the non-SAP systems for data exchange, as shown in Figure 1. The SAP data can be read through designed BAPI (Business Application Programming Interface), the middle layer is responsible for calling the data through JCO, and make the queried

data transfer to the entity JavaBeans in the form of type security using Java Proxy, then the entity JavaBeans take charge to produce a series of SQL statements and store the data to the non-SAP systems' background databases or store the data in the form of xml.

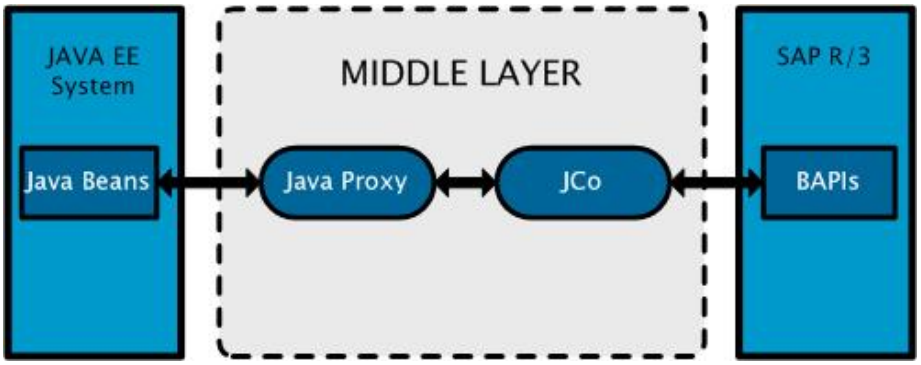

Figure 1. The Middle Layer Model

The data classification results to design different BAPI interfaces:

The first type of data: BAPI has no input parameters. Procedures for this category of data synchronization are clearing up the counterpart tables in the Java EE platform databases first, and insert the queried data from SAP one by one into the Java EE platform databases.

The second type of data: BAPI has input parameters: start-up time, termination time. The synchronization programs automatically synchronize daily based on the queried records' establish time, and the synchronization procedures do not clear Java EE databases' historical data, only incrementally synchronize.

2.4 Connect to SAP Using JCo

JCo is a high-performance, JNI-based middleware for SAP's RFC (Remote Function Call) protocol. JCo allows to build both client and server applications. JCo supports two programming models for connecting to SAP: direct connections, and connection pools. These two models also can be combined in one application. JCo's ability to use connection pools makes it an ideal choice for web server applications that are clients of an SAP system, and it also supports developing desktop applications [2]. Parts of the connection codes are shown below.

```
import com.sap.mw.jco.*;
// direct connection
try{
    JCO.Client jcoClient = JCO.createClient(
                        "600",                          // SAP client
                        "<userID>",             // userid
                        "******",               // password
                        "EN",                   // language
```

```
                    "<hostnsme>",    // application server host name
                    "21");                    // system number
                jcoClient.connect();
                }
                catch (Exception ex) {
                ex.printStackTrace();
                }
                finally{
                jcoClient.disconnect();
                }

                // connection pool
                static final String POOL_NAME = "SapPool";
                JCO.Client mConnection;
                try {
                        JCO.Pool                pool                =
JCO.getClientPoolManager().getPool(POOL_NAME);
                        if (pool == null) {
                                OrderedProperties logonProperties =
                                OrderedProperties.load("/logon.properties");
                                JCO.addClientPool(POOL_NAME,          // pool
name
                                5,
                                // maximum number of connections
                                logonProperties);
                // properties
                                }
                                mConnection = JCO.getClient(POOL_NAME);
                                System.out.println(mConnection.getAttributes());
                }
                catch (Exception ex) {
                        ex.printStackTrace();
                }
                finally {
                        JCO.releaseClient(mConnection);
                }
```

3. DATA SYNCHRONIZATION MIDDLE LAYER DESIGN AND IMPLEMENTATION

3.1 The Middle Layer Framework

According to business needs, customers issue requests, the Servlet deployed on the SAP NetWeaver server distribute the requests to entity JavaBeans, JavaBeans pass the input parameters to BAPI through JavaProxy, BAPI execute the queries using the parameters, and return the results to the entity JavaBeans through JavaProxy, then the Servlet select to deal by JSP pages, or store the data into the background database of the business systems or into the xml files. Here, Servlet play the role of controller, JSP is used to display data, entity JavaBeans responsible for handling business model. The system framework is shown in Figure 2:

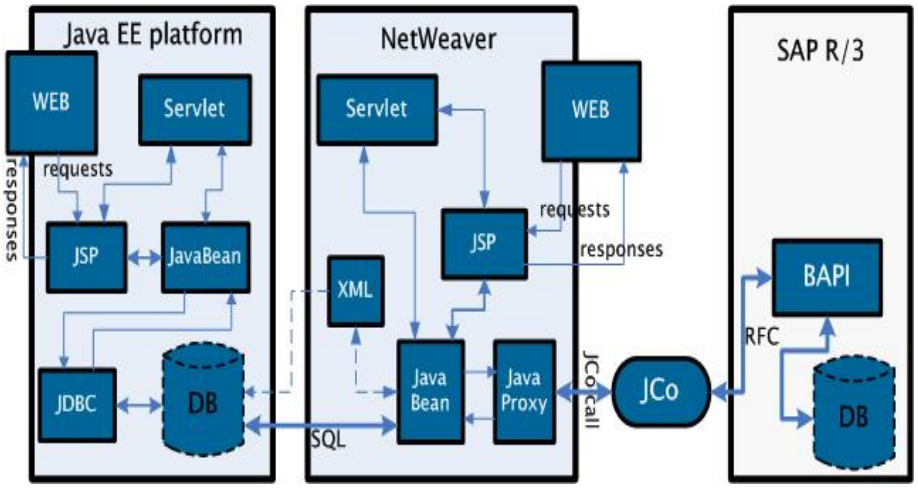

Figure 2. The System Framework

3.2 The First Category of Data Synchronization Process

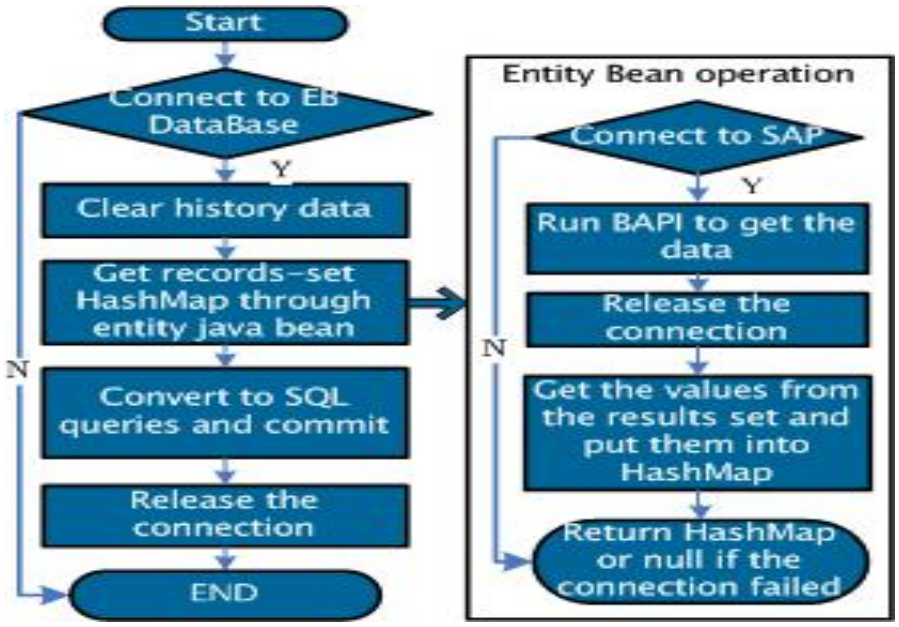

Figure 3. The First Category of Data Synchronization Process

For the first category of data, the queried results set by the implement of BAPI is stored in HashMap as a buffer, then the provisional data in the HashMap can be written as XML documents for the other heterogeneous data sources' read, or organized as SQL sequences and inserted into the heterogeneous business database directly. Process is shown in Figure 3.

3.3 The Second Category of Data Synchronization Process

This kind of data synchronization procedure is incremental synchronization which requires discriminate the queried records' creation time and the business needs, the records will be inserted into business database tables, or updated to the database tables by the primary keys. The insert operation will be changed to update when there are duplicate primary keys. If the query records' creation time is null, these records will be inserted into the business tables, and similarly, changed to update operation when the primary keys duplicate. Procedure is shown in Figure 4:

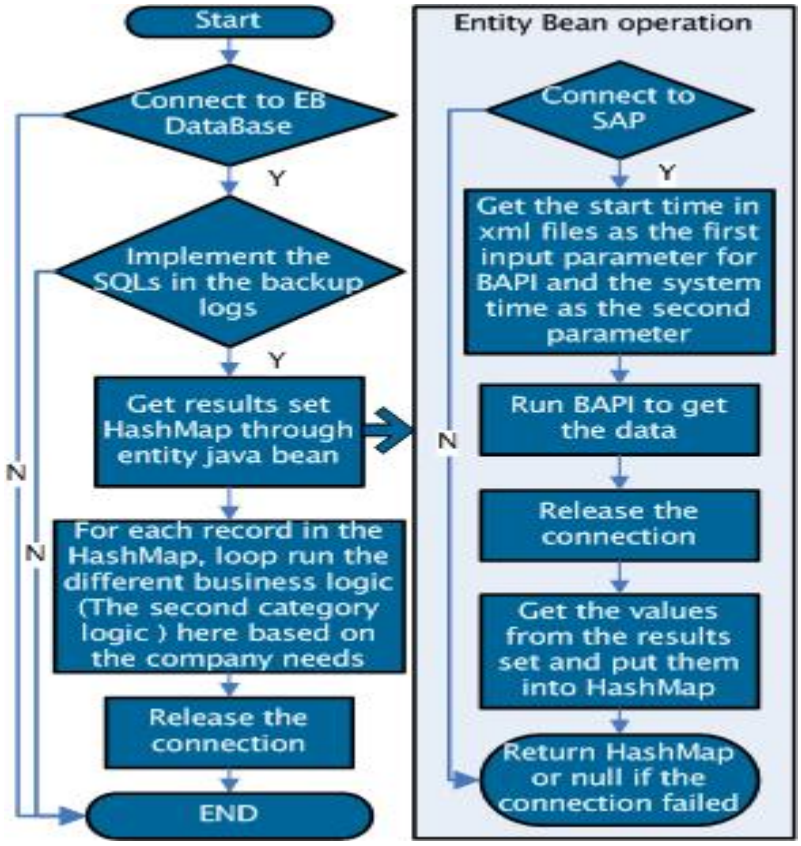

Figure 4. The Second Category of Data Synchronization Process

Based on the creation time of the data in the ERP system, BAPIs query the data from SAP conforming to the business needs, and in accordance with each record in the results set, loop run the second category logic. The second category logic is shown in Figure 5:

Definition:

ST: records query start-time // BAPI input parameter

ET: records query end-time // BAPI input parameter,

//the server's system date

CT: creation time //creation time of the main data in ERP

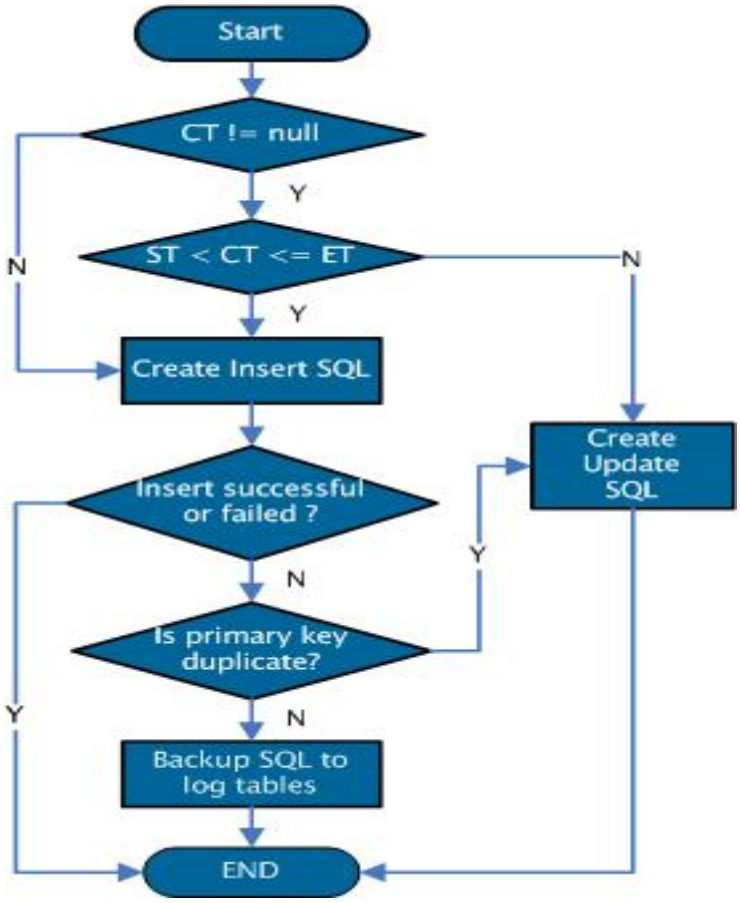

Figure 5. The Second Category Logic Process

3.4 Data Synchronization Time Nodes Storage

During the second category of data synchronization, the system time must be recorded after each operation and the recorded time will be as the start–time parameters to be inputed into BAPI in the next synchronous operation. Recording each BAPI implementation's system time in the following format:

```
… …
    <!--salers-->
        <bapi name="XSHOU_SL" date="2005-12-01" />
        <!--consignment sign-->
        <bapi name="FHUO_CS" date="2005-01-01" />
        <!--standard bom-->
    <bapi name="BPEI_STB" date="2005-12-01" />
     <!--variant bom-->
```

```
<bapi name="BSHI_VC" date="2006-12-11" />
 <!--material data-->
<bapi name="WLIAO_MD" date="2006-12-11" />
<!--customer main data-->
<bapi name="KHU_CMD" date="2006-12-11" />
… …
```

3.5 Field Mapping and Type Security

Heterogeneous systems using the relational data tables may be inconsistent with the SAP-data field's name, type or length. For the field name, use the XML document to establish the mapping relationship between the two systems:

```
… …
    <sapbapi name="DATA_BAPI_VC"> //BAPI interface name
                    <H name="FN_EB"/>
//destination database field name
                    <S name="FN_SAP"/>
//SAP-data field name
                    <H name="VW_EB"/>
                    <S name="VW_SAP"/>
    </sapbapi>
… …
```

For the field data-types, if the two system's field data-types are compatible, use JavaProxy for data-type conversion. If the data-types are not compatible, it will need to amend the e-business system's corresponding table's field data-types, and expand the length of the field to let it be able to receive the data from SAP.

3.6 Data Synchronization Logs Audit

In order to ensure the reliability of data synchronization, the middle layer records detail information of data synchronization procedures for the data calibration in the future. We use the open source component Log4J to detailed record the every operation step information. Log4J's related configuration information is shown below:

```
… …
 #root records log's level is INFO, the log information below the level is going to
 #be neglected, define an Appender named logConsole for the root recorder
log4j.rootLogger=INFO,logConsole
 #define a sapEB recorder, if there is no log level, it will inherit the root
 #recorder's level, define an Appender named logConsole for sapEB recorder,
 #sapEB is going to inherit the root recorder's Appender
log4j.logger.sapEB=,logFile
#define an Appender named logConsole and its type is ConsoleAppender
```

```
log4j.appender.console=org.apache.log4j.ConsoleAppender
#logConsole Appender's Layout is simplelayout
log4j.appender.console.layout=org.apache.log4j.SimpleLayout
#define an Appender named logFile and its type is RollingFileAppender
log4j.appender.logFile=org.apache.log4j.RollingFileAppender
log4j.appender.logFile.MaxFileSize=10MB
log4j.appender.logFile.MaxBackupIndex=2
#define logFile Appender's output path and its filename
log4j.appender.logFile.File=E:\\dataSyn\\sapEB\\ dataSyn.log
#using patternlayout as logFile Appender's layout
log4j.appender.logFile.layout=org.apache.log4j.PatternLayout
#define the output format of the logFile
      log4j.appender.logFile.layout.ConversionPattern=%d {yyyy-MM-dd
  HH:mm:ss} [%c]-[%-5p] %l "#" %m%n%n

  … …
```

Add the configuration file information to the initializing servlets, then when NetWeaver server starting deploying, it will load the initialization servlets first, thus completing the Log4j environment configuration.

4. SYSTEM ANALYSIS

SAP Java Connector is SAP's Java middleware, the SAP Java Connector allows SAP customers and partners to easily build SAP-enabled components and applications in Java. JCO supports both inbound (Java calls ABAP) and outbound (ABAP calls Java) calls in desktop and (web) server applications [2]. Java Proxy encapsulates JCO call to ensure data type security, and improve the client-end's system stability.

Data classification makes each distributed data node decide what data to synchronize according to needs and achieve local data autonomy.

The whole middle layer is developed in SAP NetWeaver Developer Studio development platform, and is deployed on the NetWeaver server, so that the whole framework's development, deployment and management are very simple and efficient, and the framework is very reliable and robust.

5. CONCLUSIONS

Based on enterprise SAP ERP's implementation and integration cases, the paper discussed in detail how to achieve the SAP system and other heterogeneous business systems' integration, proposed to use data exchange middle layer to allow developers and end-users in a transparent manner using the SAP data.

The middle layer supports heterogeneous data sources' data synchronization - transferring data between SAP and other heterogeneous data sources such as sybase, oracle, mysql and so on. The middle layer is data synchronization reliable - distributed heterogeneous system nodes using the main data consistent with the SAP main data. The middle layer is local data nodes autonomy - each node will be able to decide on their own what is acceptable data, and how to access and update the data nodes. The middle layer is centralization data management facile - the maintenance of distributed data nodes is conveniently.

In conclusion, the program is stable, flexible, user-friendly and efficient in the enterprise SAP ERP's implementation and integration.

REFERENCES

1. D. Smith, L. O'brien, K. Kontogiannis, and M. Barbacci, *Enterprise Integration*. http://www.sei.cmu.edu/news-at-sei/columns/the_architect/2002/4q02/architect-4q02.htm(Accessed July 8, 2007).
2. G.S. Thomas, Developing Applications with the "SAP Java Conne-ctor" (JCo), *ARAsoft GmbH* (2001-2002). www.ARAsoft.de
3. Anonymous, *SAP Java Connector*. http://help.sap.com/saphelp_47x200/helpdata/en/6f/1bd5c6a85b11d6b28500508b5d5211/frameset.htm (Accessed December 26, 2006).
4. K. Kessler, P. Tillert, and P. Dobrikov, *JAVA Programming with the SAP Web Application Server* (Oriental Press: Beijing, 2005).
5. M.N. Huhns, and P.S Munindar, Service-Oriented Computing: Key concepts and Principles, *IEEE Internet Computing*. Volume 9, pp.75-81, (2005).
6. Anonymous, *Eye on integration*. http://www.sei.cmu.edu/news-at-sei/Columns/eye-on-integration/eye-on-integration.htm (Accessed December 20, 2006).
7. M. Gangwani, *Java SAP R/3 Integration*, Persistent White Paper (November 2004).