UML MODELING WITH JDEVELOPER 10.1.3

Ann Horton, Oracle Corporation

Introduction

Have you tried the UML Modeling capabilities of JDeveloper 10.1.3 yet? If not, you definitely should! JDeveloper 10.1.3 provides greatly enhanced UML modeling capabilities including: improved layout capabilities, sequence diagrams, use case diagrams, more complete activity diagram notation, and more robust Java class and database diagrams. This presentation will examine each diagram, and explain the diagram's purpose, syntax, and how the diagram supports the software development lifecycle. These capabilities are easy to use. You should definitely give them a try – even if you haven't used JDeveloper or you don't use Java! Yes, these tools are useful even if you don't use Java.

Background

JDeveloper 10.1.3 was released into production in January 2006. It is a very powerful development tool, with an extensive list of new features including:

- Greatly improved usability with a refined look and feel throughout the tool
- JavaServer Faces (JSF) support and expanded ADF DataBindings
- Support for J2SE 5.0, J2EE 1.4, and EJB 3.0 (Toplink)
- Numerous new Web Services Features including JAX-RPC, SOAP 1.2, WSDL Editor
- Enhanced ADF Swing with JGoodies Layout Management
- Numerous new features for database management.
- Improved SCM integration.

JDeveloper continues to offer "Productivity with Choice". If you haven't heard – JDeveloper is now "free".

JDeveloper UML Diagram Types

JDeveloper 10.1.3 supports the following major diagram UML diagram types:

- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Database Design Diagrams
- Specialized Class Diagrams
    - Java Class Diagrams
    - Business Components Diagrams
    - EJB Diagrams.

Business process diagrams are also available using the JDeveloper BPEL Designer - a "plug-in" to the base JDeveloper.

The Unified Modeling Language (UML) standard specifies the syntax of these diagrams. However, it does not specify a software process. These diagrams can be used with any software process that you like. You will find these diagrams useful throughout your software development process.

Getting Started

The objective of this presentation is to get you started using the diagramming features of JDeveloper.

Download and Install JDeveloper 10.1.3

If you are not already using JDeveloper 10.1.3, then your first step is to download and install JDeveloper.

1. Go to OTN, http://www.oracle.com/technology/products/jdev/index.html, find JDeveloper 10.1.3, and select the bundle which includes all the UML diagrammers – currently called the "J2EE Edition". Be sure to download the *Complete Install* which includes the Java Development Kit (JDK 5.0).
2. Unzip the downloaded file into a <JDev Home> directory – for example c:\jdev. Now you are ready to use the tool. JDeveloper does not install under the Universal Installer. Be sure to check the Release Notes and Installation notes.
3. To invoke JDeveloper, click on <JDev Home>\jdev\bin\jdev.exe and the tool will come up.
4. Patches are regularly released for JDeveloper. So it is always a good idea to check for any updates applicable to your downloaded version. From the Help menu, select "Check for Updates" and follow the dialog windows to check for any updates, and download/install any that are applicable. Note that you will need an OTN account to download the updates.

Create an Application Workspace and a Project

To develop diagrams within JDeveloper, you must first create an Application and a Project within that Application.

1. In the Applications Navigator, right click *Applications* and choose *New Application* from the context menu. As shown in Figure 1, be sure that "No Template (All Technologies)" is selected.

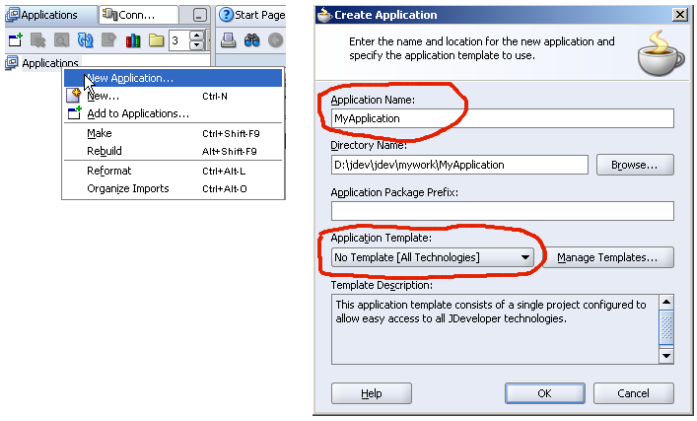

Figure 1 – Create New Workspace

2. Next create a new Project within your Application.

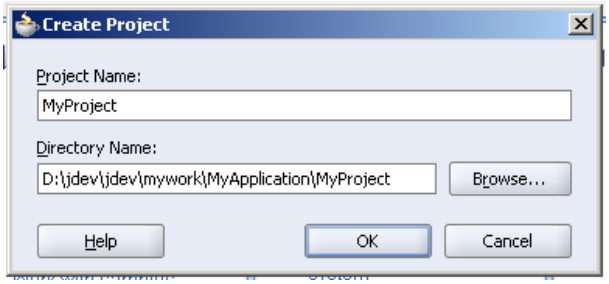

Figure 2 – Create Project

3. Now you will see your Application and Project in the Applications Navigator as shown in Figure 3.

Figure 3 – Applications Navigator with Application and Project

4. Click the *Save All* button to save your changes.

Create a Diagram

Now you are ready to create a Diagram in the project that you have built.

1. Right click your Application, and select *New* to bring up the *New Gallery* and you will find the available diagrams under the General Category:

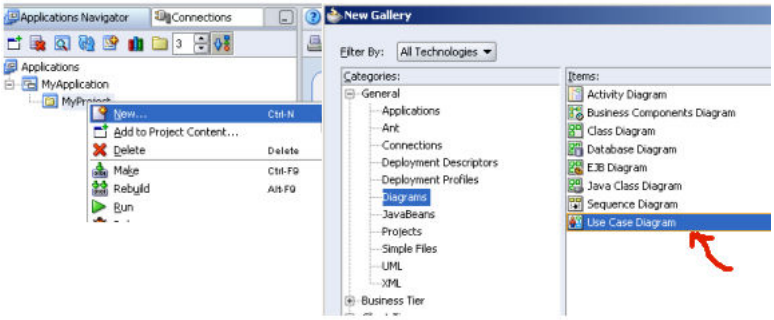

Figure 4 – New Gallery

2. Choose the diagram type. For example select *Use Case Diagram*.
3. The *Create Use Case Diagram* dialog window will appear (Figure 5). Name your diagram and define a package to contain the diagram. The package is just a folder to hold the files associated with the diagram within the Application and Project directories.

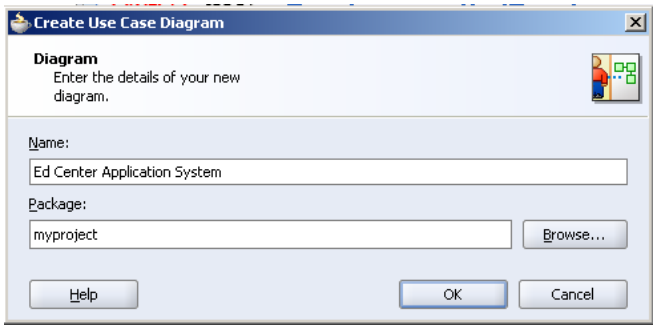

Figure 5 – Create Diagram

4. After you complete the Create Diagram dialog, a blank diagram canvas (#1) will be displayed ready for you to drag diagram components onto as shown in Figure 6. The window configuration is the same for all diagram types but the Component palette is specific to the selected diagram type.

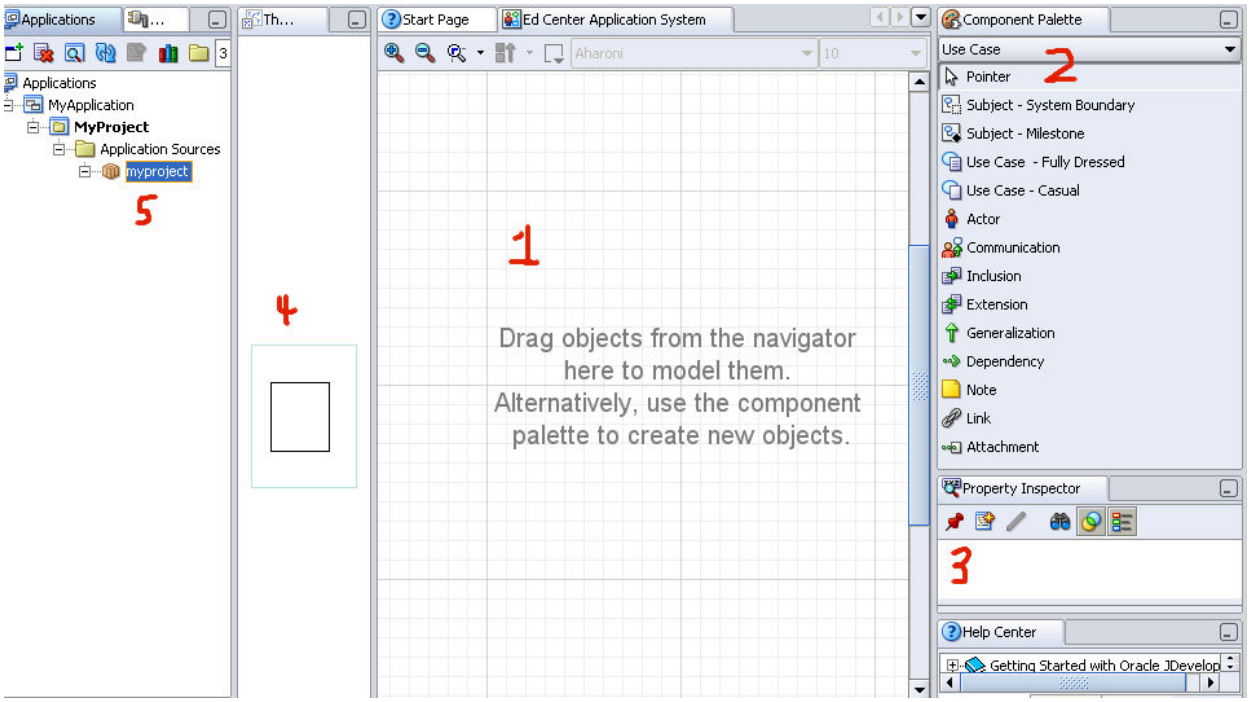

Figure 6 – Diagram Canvas and Associated Windows

The Component Palette (#2) contains elements that can be added to the diagram by dragging them onto the canvas. The Property Inspector (#3) will display properties of a selected diagram component. The View Locator window (#4) indicates the location of the current diagram view and supports navigation around the diagram. As components are added to the diagram, they will appear in the Applications Navigator (#5).

Now that you understand how to create a diagram, let's look at each of the key UML Diagrams. We will start with Use Case Diagrams.

Use Case Diagrams

Use Cases define the functions to be performed by a system. They are similar to the business functions in CASE Method's Function Models. A Use Case is a *Function to be performed by the System from the User's Perspective.* A Use Case "name" is a verb phrase – for example:

- Browse Course Schedule
- Enroll the School
- Register for a Course Session
- Authorize Credit Card Charge.

A Use Case is initiated either by an Actor or the System. An Actor is anything outside of a system that must interact with the System – for example a Role Played by a Person, or an External System.

Figure 8 shows a sample Use Case Diagram. Each Oval is a Use Case, and the people figures are the actors.

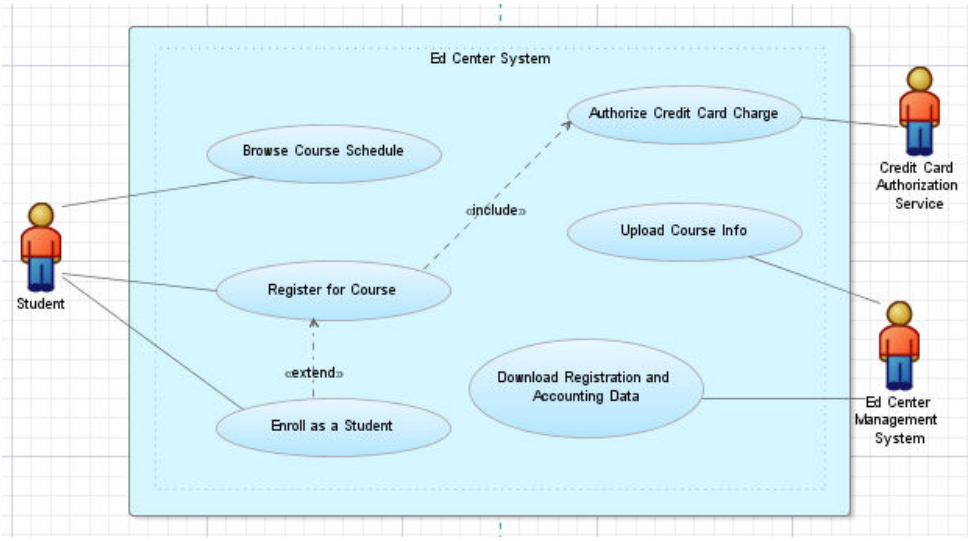

Figure 8 – Sample Use Case Diagram

Two or more use cases may include the same steps. There are two special relationships which support re-use of use case logic:

- An <<includes>> relationship indicates that the second use case is *always* invoked by the first use case.
- An <<extends>> relationship indicates that the second use case may optionally invoke the first use case.

Figure 9 shows the JDeveloper Component Palette for Use Case Diagrams.

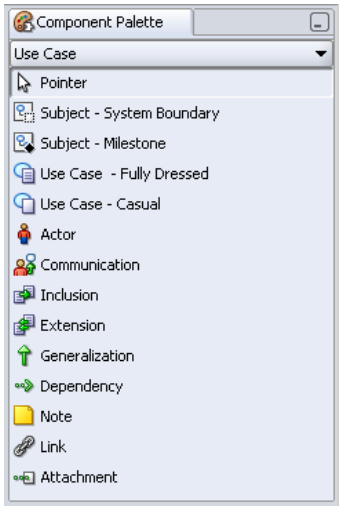

Figure 9 – Use Case Diagram Components

A Use Case is a *Textual Specification* that defines the series of steps (or the Flow of Events) required to accomplish the goal of the use case. A Use Case will consist of multiple scenarios:

- The primary scenario ("Happy Path") flow defines the steps when everything works perfectly
- One or more alternate exception flows that specify the steps and actions when everything is not perfect.

Developing the exception flows is the most challenging part of defining a use case. There are three ways to document alternate workflows:

1. Write each alternate use case as a separate scenario – recommended for longer alternate flows.

2. Write extension steps tied to main scenario steps – recommended for short variations.
3. Write IF statements throughout the main scenario – not recommended.

Use Cases were invented by Ivar Jacobsen in the late 1960s, and adopted by the OO community in the late 1980s/early 1990s. The UML standard only defines the syntax for a Use Case diagram. There are no standards that define the textual specification of a Use Case – so a variety of document templates exist.

JDeveloper provides two Use Case textual templates:

- A "full-dressed" Use Case has numerous formal sections.
- A "Casual" Use Case specification has fewer sections, and tends to be more narrative in nature.

These templates are accessible by clicking a Use Case oval from the diagram. This text may also be published in HTML format and made available to your users.

Activity Diagrams

An Activity Diagram is a dynamic model of a business or software workflow. It presents the sequencing of activities with support for parallel activities, and emphasizes the flow of control from one activity to another activity. UML activity diagrams combine ideas from several techniques including event models, state diagrams, workflow modeling, business process modeling, and Petri Nets. An activity diagram is useful to explore the logic of: a business process, software processes, a single use case, or several use cases.

An Activity (or Activity State) is a state of doing something and is represented with a Verb Phrase. A transition is a flow of control from one activity to another.

Activity Diagrams depict *what* happens not *who does what* in a Business Process or what Class or Program does what in a Software Workflow. Swimlanes provide a capability to group activities by who performs them.

Swimlanes are horizontal or vertical Zones of a Diagram. Each Zone represents the responsibilities of a particular Software Class or a particular Organizational Entity.

Figure 10 shows a sample Activity Diagram.

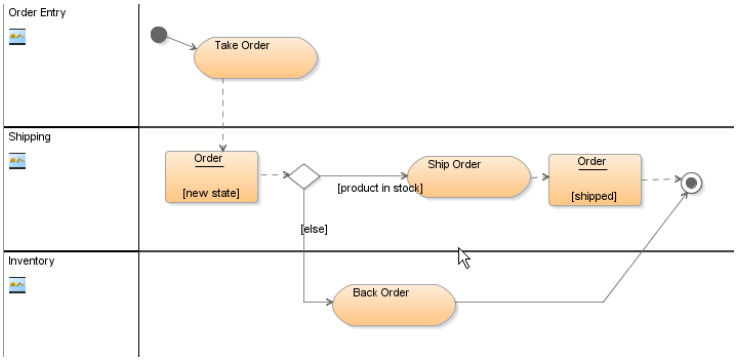

Figure 10 – Activity Diagram

Figure 11 shows the JDeveloper Component Palette for an Activity Diagram.

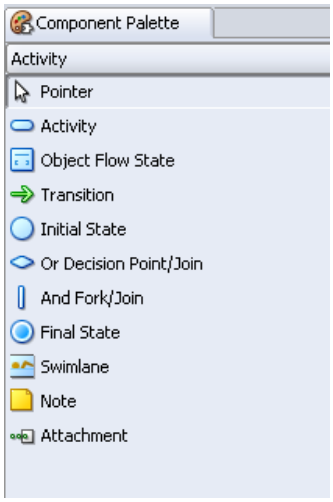

Figure 11 – Activity Diagram Components

Class Diagrams

Class Diagrams are the heart of object-oriented analysis and design. A class is a "thing" and has a name, operations, and attributes. UML class diagrams show the classes of a system, their interrelationships - including inheritance, aggregation, and association. During the software development process, Class diagrams are developed at a variety of levels of detail, and for a variety of purposes.

A business class diagram might be developed to show the major business classes or objects within an enterprise. A more detailed analysis-level class diagram might be developed to define the logical attributes and operations required from a new system. An implementation-level class diagram would show specific Java classes in a system implementation, and the interrelationships between those classes.

JDeveloper supports the development of the following class diagrams:

- UML class diagrams with any level of detail
- 3 specialized Class Diagram Types:
  - Java Class Diagrams
  - Business Components Diagrams
  - EJB Diagrams.

Samples of each of these diagram types will be provided during the presentation.

JDeveloper supports the transformation of certain modeled elements into other modeled elements. These elements are as follows:

- UML classes and interfaces can be transformed to Java classes and interfaces
- Java classes and interfaces can be transformed to UML classes and interfaces
- UML classes and interfaces can be transformed to Business Components entity objects.

Transformations can be invoked on individual elements, groups of elements on a diagram, or on entire diagrams. If you want to implement in Java a design that has been modeled using UML classes and interfaces, you can transform your modeled UML classes into modeled Java classes

Sequence Diagrams

UML Sequence Diagrams model the time sequencing of messages within your system and show how objects interact with messages. Sequence diagrams are widely used for software design and for illustrating design patterns. JDeveloper provides the capability to develop sequence diagrams, and also provides the capability to connect the Sequence Diagram modeler to the Debugger to visually debug an application.

As shown in Figure 12, a sequence diagram shows a set of objects horizontally across a page. Each object has a dashed vertical lifeline. An activation bar on the lifeline shows when that object is active. A Message is represented by a solid directed line URL and a Return is represented by a dashed directed line Home Page. Each Message and Return may be named.

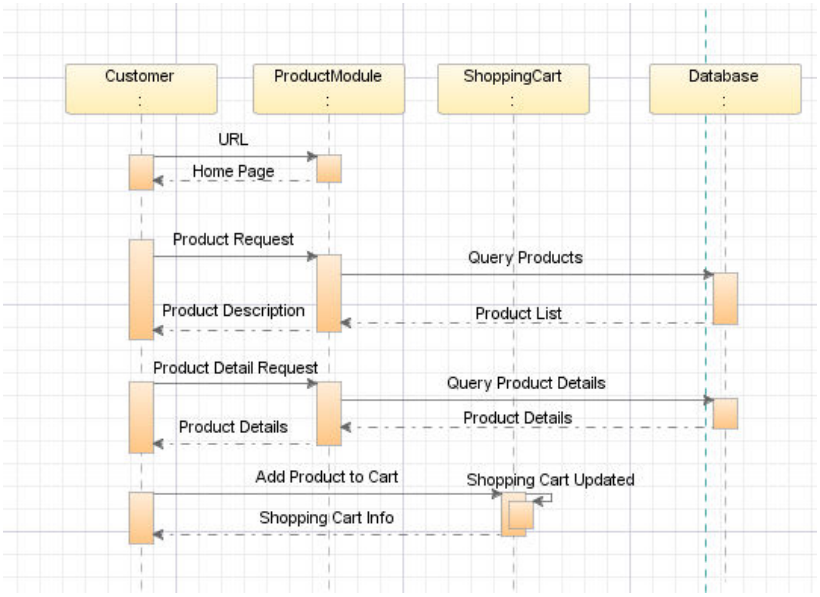

Figure 12 – Sequence Diagram for Product Catalog

The JDeveloper Component Palette for sequence diagrams is shown in Figure 13.

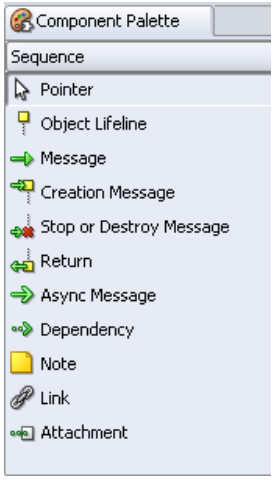

Figure 13 – Sequence Diagram Components

Database Diagrams

JDeveloper 10.1.3 provides a powerful facility for database design and development. A Database Diagram can be developed manually, and then physical tables and other database objects can be generated from that diagram. The JDeveloper 10.1.3 database diagrammer now supports the creation, generation, capture, and visualization of the following objects (in addition to tables, columns, and keys): Views, Sequences, Synonyms, Indexes, PLSQL Packages, Procedures, Function, User-Defined Types, XML, Spatial, and Media System-Defined Types.

JDeveloper also supports design capture from an existing database. A Database Diagram can also be easily created from an existing Oracle database. JDeveloper 10.1.3 can now additionally import and model objects from MySQL, MS SQL Server, Sybase, IBM DB2, and IBM Informix in addition to Oracle databases.

Figure 13 shows the dialog for creating JDeveloper database objects from the OE schema of an Oracle database.

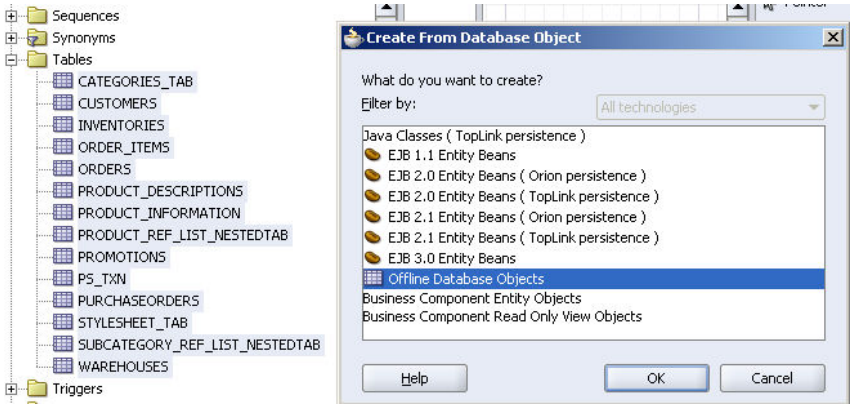

Figure 13 – Reverse Engineering Wizard

In this example, we selected Offline Database Objects, but we could have created other JDeveloper object types from this physical database schema. The resulting JDeveloper Database Diagram is shown in Figure 14.

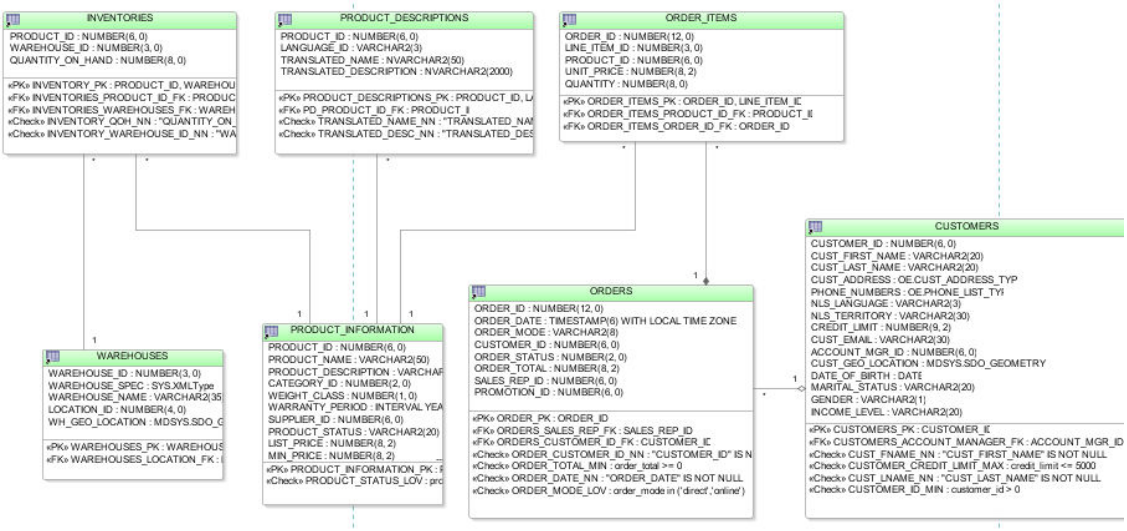

Figure 14 – Database Diagram for the OE Schema

JDeveloper now offers an UI for visually comparing the offline database objects with the online schema, allowing changes to be selectively applied instead of an all-or-nothing approach to database reconciliation. ALTER SQL scripts are automatically generated when required.

Summary

The diagramming capabilities of JDeveloper 10.1.3 have been greatly improved. JDeveloper now provides a powerful set of UML modeling capabilities including use case diagrams, activity diagrams, class diagrams, sequence diagrams, and database diagrams. These capabilities are easy to use – you just need to download JDeveloper and unzip the distribution, and you are ready to diagram.

A picture is worth a thousand words! You should definitely give these capabilities a try!

About the Author

Ann Horton is a Senior Principal Consultant with Oracle's Advanced Technology Services (ATS) Fusion Middleware Group. She currently works with JDeveloper, ADF Faces, Application Express, BPEL, and Oracle Portal. Ann has over 17 years of experience applying Oracle technologies to solve enterprise business problems and is a Sun Certified Programmer for the Java 2 Platform. Ann has helped numerous organizations apply Oracle technologies, and has taught graduate level university courses in Java, object-oriented technology, and database technologies. She is co-author of the book Professional Oracle 8i Application Programming with Java, PL/SQL, and XML. Ann is a frequent presenter at RMOUG, ODTUG, and IOUG-A. Ann is internationally known for her expertise in data modeling, database design, business modeling, and Oracle's Designer product.

References

For more information on this subject, see the following:

1. The JDeveloper 10.1.3 Help System
2. OTN: http://www.oracle.com/technology/products/jdev/
3. JDeveloper discussion forum: http://www.oracle.com/technology/discussionforums/jdev.html
4. JDeveloper RSS news feed: http://www.oracle.com/technology/products/jdev/temp/whatisrss.html
5. Oracle's Monthly Java Newsletter: http://www.oracle.com/technology/tech/java/newsletter/archive.html
6. UML Distilled, Third Edition, A Brief Guide to the Standard Object Modeling Language, by Martin Fowler, Addison-Wesley, 2004, ISBN 0-321-19368-7