Database access control within a Java application

Alain Plantec, Vincent Ribaud, Philippe Saliou

Database access control within a Java application

A.Plantec, V.Ribaud and P. Saliou
EA2215, Département d'Informatique, Faculté des sciences, 29285 Brest Cedex, France
E-mail:{alain.plantec,vincent.ribaud, philippe.saliou }@univ-brest.fr

Abstract

The purpose of the work described in this article is to use database security mechanism as a data access control model in a Java application, developed through an UML-based process. This work relies on the CASE tool Designer from the Oracle company and its associated user and security policy.

The three axis of this work are the following ones :

- Introducing class diagrams towards Designer analysis and design models (entity-relation ship diagram, design diagrams).
- Using Designer features to design user and security policy, and generate database schema.
- Reconciling object-oriented and relational worlds through the use of TopLink, an object-relational mapping tool and with the exploitation of a dedicated generator, intended to make the security policy and related information available to the Java application.

1. Introduction

UML-based software development is not primilary intended to relational database design and implementation. However, in most object-oriented applications, data are still handled through a relational database management system (RDBMS). The figure 1 presents typical architectures :

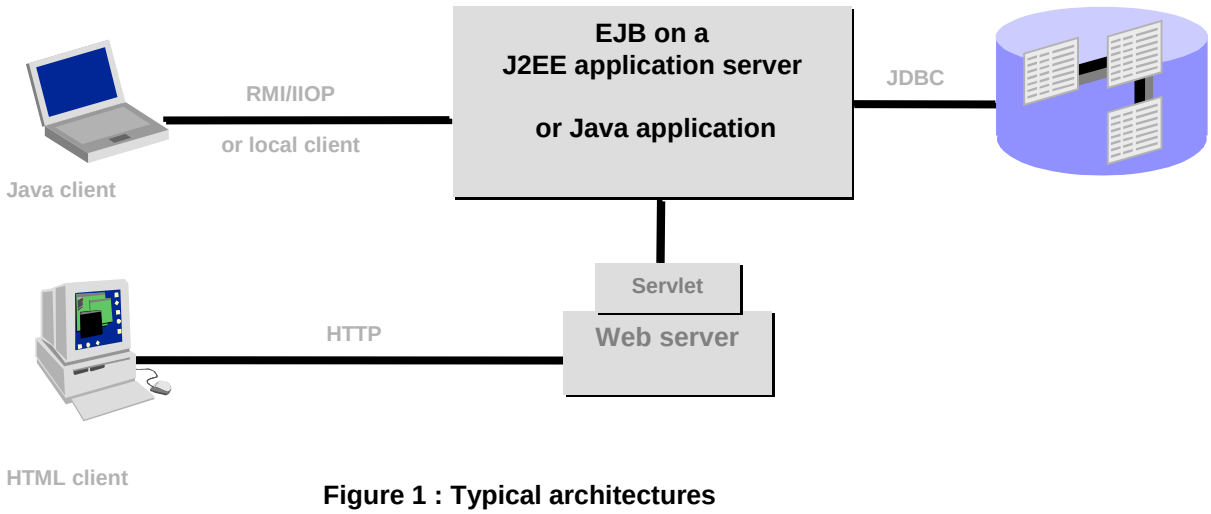

Figure 1 : Typical architectures

Object-oriented applications and database information must be bridged. Bridging is usually accomplished through the use of an Object/Relational Mapping Product such as TopLink.

The development of an object-oriented application using a relational database is a process which is divided in two subprocesses : analysis-design-implementation of the object-oriented part and design-implementation of the relational database. Unfortunately, UML-based tools offer very poor database design and implementation features. Despite the fact that UML profiles (and especially Database profile) aim to enhance these features in UML tools, relational CASE tools are richer and more mature tools.

Database design and implementation includes user and security management, especially authentification mechanism, quotas, privileges and roles. Oracle Designer is a CASE tool which offers high-level user and security management models which are *in fine* translated in the database schema.

The paper describes that the two subprocesses should be supported with two different tools : an UML-based tool for the object-oriented part and Oracle Designer for the Oracle database part (including the access control policy), with the use of an Object/Relational mapping to reconcile both parts. Mapping UML class diagrams with Designer diagrams can be accomplished in different ways : with Oracle UML-based tools or with re-engineering techniques. Inside the database world and tools, the security policy is

designed and implemented in the database schema. Some useful database security model and information related to should be reintroduced in the object-oriented world. This is accomplished through a dedicated generator. This generator is built with Eugene, a STEP-based framework, that is intended for the building of application generators.

The article is organized as follows. Section 2 presents the Oracle's user and security model. Section 3 presents class diagrams coupling with the CADM method (Oracle Designer method) and reconciliation using TopLink. Section 4 describes the tools integration. Section 5 summarizes related work and perspectives and is followed by a conclusion.

2. How security domain is managed in Designer and Oracle ?

The database administrator defines the names of the users allowed to access to a database.

A schema is a named collection of objects such as tables, views or stored procedures. The schema is owned by a database user and has the same name as the user and therefore username and schema are synonyms.

Database security can be classified into two categories : system security and data security. System security covers access and use of the database at the system level, such as username and password, disk space allocated to the users, and system operations allowed by the user. Database security covers access and use of the database objects and the actions that those users can have on objects.

A security domain defines the settings that apply to the user. The complete features covered are depicted in figure 2.

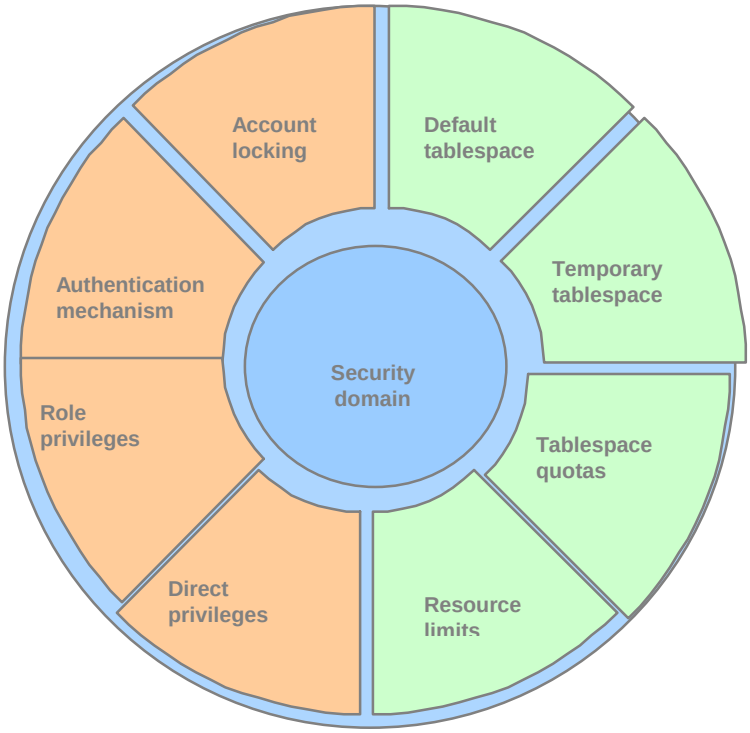

Figure 2 : Database security features

In the database access control perpective, authentification mechanism, account locking, direct privileges and roles are useful.

- Authentification mechanism : a user can be authentificated by one of the following : database, operating system, network.
- Account locking : accounts can be locked to prevent a user from logging on to the database
- Direct privileges : privileges are used to control the actions a user can perform in a database
- Role privileges : a user can be granted privileges indirectly through the use of roles

Each user (the grantor) can grant any privileges to a user (the grantee) on any object belonging to his/her schema. Privileges can be granted with the grant option, which means that the grantee can also grant the privileges to another user. Grantors can revoke privileges from only those users whom they have granted privileges. Revoking privileges will cascade when privileges were given with the grant option.

Roles are named groups of related privileges that are granted to users or other roles. Using roles simplifies privilege management. Rather than granting the same set of privileges to several users, the privileges are granted to the role, and then the role is granted to each user. The role management is dynamic. If the privileges associated with a role are modified, all the users who are granted the role automatically and immediately acquire the modified privileges. Roles can be enabled and disabled to turn privileges on and off temporarily. Within roles, object privileges can be revoked without causing cascading revokes.

Fine-grained access control allows to implement security policies with packages of functions and then associate those security policies with tables or views.

The implementation of fine-grained access control is made through dynamic modification. When a user accesses an object (either directly or through a subquery) that has a security policy attached to it, the RDBMS automatically consults the package that implements the policy for that view or table. The policy returns a predicate (access condition) that is appended to the query. The statement is then parsed, optimized, and executed.

Working at the SQL level, all these features are available through an Oracle tool called Security Manager. The same features can be specified through the Oracle CASE tool, Designer.

Designer and the associated CADM method

Oracle Designer is an extremely powerful integrated CASE tool. It allows the whole building of an Information System all along the phases of the software life cycle. To this end, it relies on a unique common repository stored in an Oracle database.

The development approach relies on CADM (CASE Application Development Method), a waterfall process (Analysis->Design->Build->Implement->Production)) by Paul Dorsey and Peter Koletzke [1]. This approach is a derivative of the Case*Method by Richard Barker [2].

CADM belongs to the family of systemic methods. The data and processings have first to be separately modelized, and then coupled to constitute a unique and integrated system. The building of the system gets through different abstraction levels: analysis, design and implementation.

An important feature is the definition of usages : which entities (tables) are used by functions (modules) and how functions (modules) use entities (tables) i.e. does the function (module) Create, Retrieve, Update, or Delete instances of the entity (table) ? The CRUD matrix is a two-dimensional chart that summarizes usages between functions (modules) and entities (tables).

Defining usages is a part of security policy, because it defines data access control inside application modules.

During the general analysis phase, the requirements capture becomes elaborated, detailed and reshaped through a function hierarchy and an Entity/Relationship data model.

The function hierarchy is the model proposed by Designer to analyze processings. Processings in the information system are hierarchically divided into a set of activities known as functions. Therefore a function is a more or less important activity which can be either automatized or manual.

During the detailed analysis phase, the function hierarchy is refined and completed: entities usages as well as attributes usages are defined for each function while cross-reference controls are performed between data and functions.

At the beginning of the design phase, functions are mapped to the application modules and the E/R model is mapped to a relational model. A module is structured into modules components, which can be *in fine* translated either Oracle 4GL constructs (Forms block), either in a package of Java classes, or a set of Web pages.

During the design phase, theses modules and database objects are refined and completed: tables usages as well as columns usages are defined for each module while cross-reference controls are still performed.

Security and access control is designed during this phase.

The build phase involves two areas : the database and applications. Database building is a straightforward SQL generation operation. Application building is out the scope of this paper, except for the design and implementation of stored procedures. In this particular case, Designer facilitates modules editing, code generation and ensures the consistency of the repository.

3. Coupling class diagrams and CADM models

UML and databases

The class diagrams depict a static view of the system. This static view will lead to a static part (class and instance variables, relationships) in the object-oriented classes and a SQL schema (tables, views, constraints, ...) together with a security policy (users, privileges, roles).

UML, the Unified Process [3] and UML profiles have made popular the idea of complementing the UML models with stereotypes in order to deal with the problems which are not adressed by UML constructs. In the particular case of using a database in order to manage the persistence of objects, this leads to the following use of UML-based tools (figure 3) :

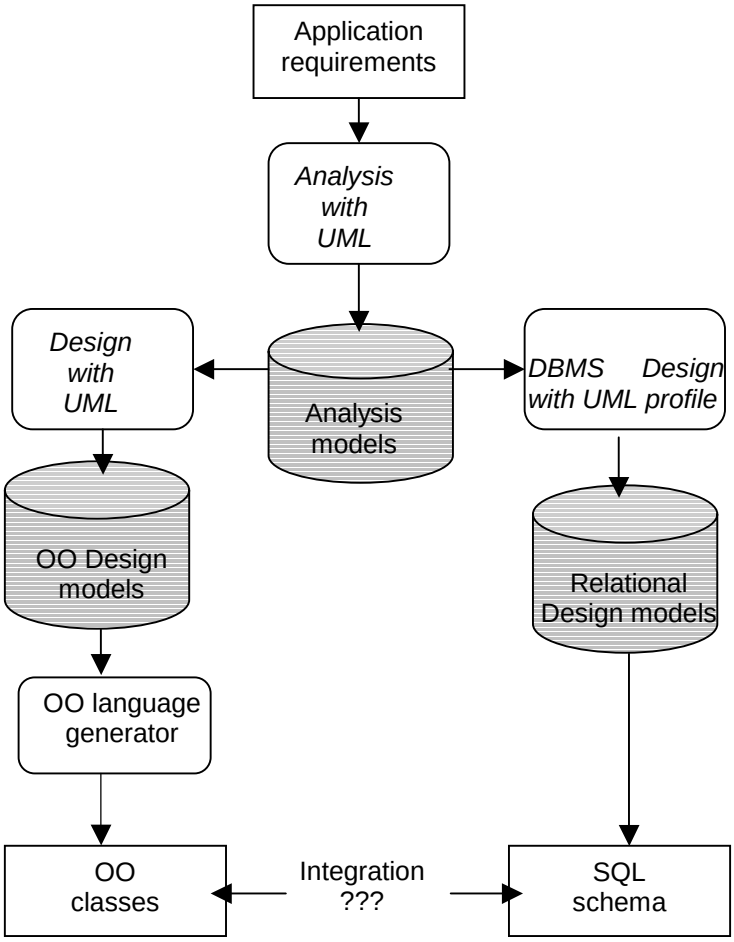

Figure 3 : Development process

An UML profile is an UML extension whick keeps intact the UML metamodel. The UML profile for database design is a set of stereotypes and marked-values attached to these stereotypes which are used to represent usual constructs in database design : domains, tables, views, primary and foreign keys, identifying relationships [4].

The integration of an object-oriented application such as a Java application with database information is one of the most underestimated problems.

Why introduce another tool of database design ?

Despite UML proselytes, enhancing UML and UML tools with a collection of database-oriented stereotypes does not lead to the power and the reusability claimed. The authors of this paper believe in “the best of breed” concept : using UML to design object-oriented applications and using proven relational methods and tools to design the database. This approach decomposes the development process into two parallel subprocesses (analysis, design, implementation).

The two subprocesses are forked at the end of the UML analysis phase (see figure 5). Class diagrams are translated in E/R diagrams. Oracle company offers (and offered) various UML tools which can be used for this translation. Translating class diagrams into a database schema followed by the re-

engineering with Designer of the database schema into relational constructs then in E/R constructs is another possible solution.

The two subprocesses are reconciled through the use of TopLink (formerly a third-party product, purchased by Oracle in 2002 and now released Oracle9iAS TopLink). TopLink is a Java-to-database integration product which includes Object-Relational mapping. TopLink provides a solution to address the complex differences between Java objects and relational databases and enables applications to store persistent Java objects in any relational database supported by a JDBC driver. TopLink includes a visual tool, the Mapping Workbench, used to map any object model to any relational schema (figure 4). TopLink Mapping Workbench creates metadata descriptors (or mappings) that define how to store objects in a particular database schema. TopLink manages the metadata in a session. The session can be exported (or imported) as a XML configuration file, called “sessions.xml”. The DTD is shipped with TopLink.

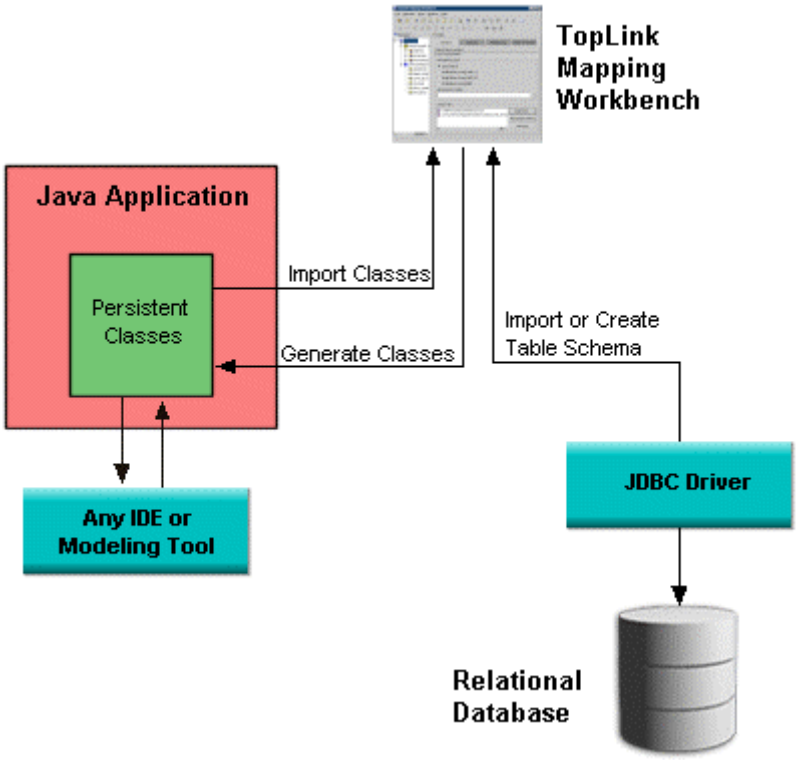

Figure 4 : The mapping process

The Foundation Library contains the TopLink API that is included and called at runtime in order to retrieve and store Java objects. The runtime is driven with the particular “sessions.xml” configuration file describing the mapping metadata of the application.

Security policy

Object-oriented applications need to manage users to control access to the data and to enable or disable application functionalities. Moreover, when using a database, the object-oriented application has to deal with database accounts.

Otherwise, there are several ways to manage database users within an object-oriented application. Sometimes an unique database account is used which owns the whole database. Sometimes each type of users corresponds to a database account. Sometimes each user of the object-oriented application has its own database account.

Limits

We are using Oracle Designer only for the analysis, design and implementation of the database (or the database-tier in a n-tier architecture). Then, we are loosing the usage definitions described at the end of section 2. The security policy that we are able to design and implement is the policy available at SQL level (users, privileges and roles) depicted at the beginning of section 2.

There is a way to minimize this drawback. RDBMSs allow the use of stored procedures. Stored procedures are subprograms written either in a dedicated procedural language (PL/SQL in the case of Oracle) or in Java. Stored procedures are executed by the RDBMS (the database-tier).

Batch modules (not interacting with the end-user) can be translated in stored procedures and usages security policy (CRUD control) can be applied to these stored procedures. Moreover, as part of the database schema, the privilege of executing a stored procedure can be granted to users and roles.

4. Tools integration

As stated in section 3, the process development is subdivided in two sub-processes.

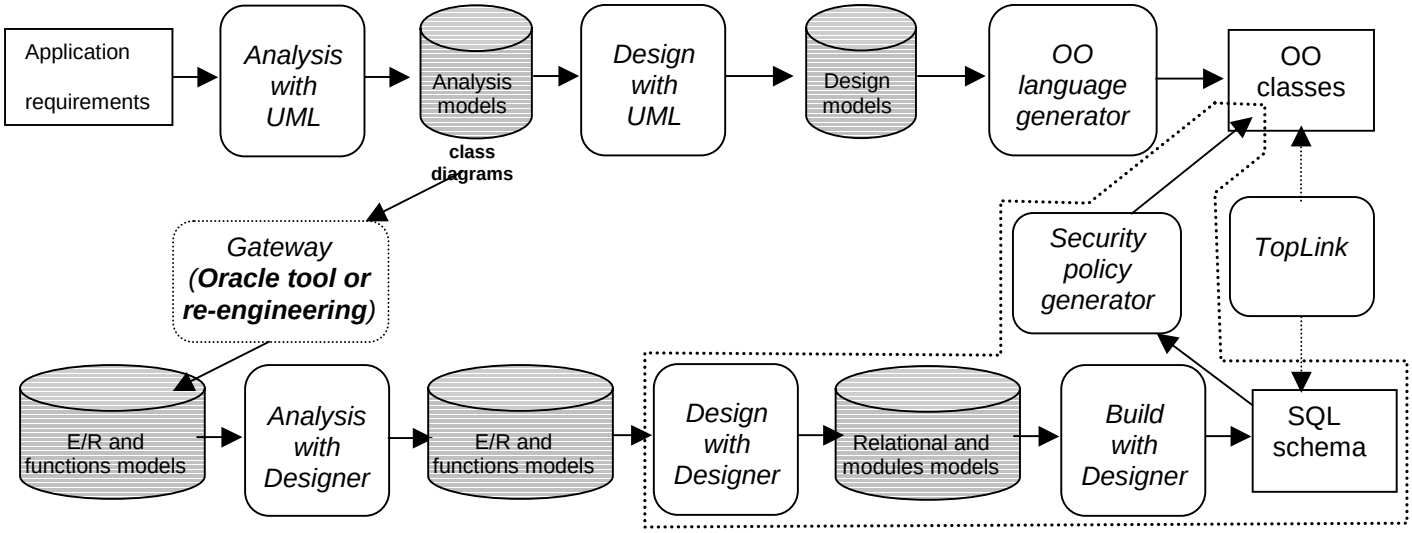

Figure 5 : The development process

Most of reconcilation is done with TopLink, but the integration of a part of the security policy should to be done.

Oracle security mechanisms define a model for managing users, privileges and roles. The different ways of using database accounts (see the discussion at the end of section 3) are some kind of design patterns for a security model. Implementing a security policy in a particular application instanciates a security model. Access to the database security model and the information related to it can be useful to the object-oriented application.

It is possible to overload UML analysis and design (and especially class diagrams) with the modelization of classes “user”, “privilege” and “role” and relationship between these classes. Then these artifacts need to evolve through the development process in order to provide a set of classes managing the security policy and access control to the database.

TopLink itself can also be used. The database dictionary (the metabase) is a database too, and mapping can be accomplished between metatables and application classes.

We use another approach. We built a dedicated generator which produces the set of security classes from the security information found in Designer repository and from the generator user’s choices regarding the type of pattern wanted for the application. This approach simplifies development process and prevents desynchronisation between object-oriented and relational parts. The security policy generator is built with Eugene, a STEP-based framework [5], that is intended for the building of application generators [6]. The scope of the generator is dotted in figure 5.

5. Related work and perspectives

This work allows database (and data) access control in Java applications using a relational database. The more general problematic of coupling Java applications with a Relational DataBase Management System is adressed and resolved with Integrated Development Environments (IDE) such as JDeveloper from Oracle or JBuilder from Borland.

JDeveloper was early designed for Java developers who were looking for ways to create applications that would interact with Oracle databases. JDeveloper evolved to Oracle9i JDeveloper which is built on Java 2 Platform, Enterprise Edition (J2EE). One of JDeveloper's major features is its Business Components for Java (BC4J) framework and associated code generators. JDeveloper uses BC4J as the primary way of handling database DML operations (queries, inserts, updates and deletes). BC4J components are built using a combination of Java and XML. XML is used to define the data, and Java is used to operate on the data. BC4J components will have both XML and Java files associated with them. The XML file holds the metadata that defines the business component, and the Java file holds the methods that implement the business component. The Java file contains accessors methods, which dynamically generate insert, update, and delete statements at runtime [7].

JDeveloper includes an UML class diagram modeler. This modeler allows to define classes and associations and to connect them with BC4J entities and BC4J associations. While BC4J elements are managing the database, BC4J and TopLink appear to be competing technologies. But neither TopLink nor BC4J address the security domain of an Oracle database and Designer remains the only high-level tool which allows to design and implement a security policy.

A first perspective depends on the future of Oracle tools. When JDeveloper will be effectively competing with Designer as a data modeling tool, security features should be supported and the work presented in this paper will loose interest. But until that time, improving security policy design (especially patterns of security model and fine-grained access control) and enhancing code generation is useful.

As stated at the end of section 3, an interesting perspective lies on the exploitation of stored procedures generated from Designer modules employing tables and columns usages. These stored procedures should be used to establish an higher level of security policy based on CRUD definitions.

As a last perspective, automatic mapping features can be envisaged through another dedicated generator. The Mapping Workbench can automatically map class attributes to a similarly named database field or from table references, but mapping Java objects and relational constructs with TopLink still remains a tedious task. It would be useful to produce automatically a "sessions.xml" configuration file stemming from class diagrams, relational models and user's choices in order to import it at TopLink runtime.

6. Conclusion

This paper has presented how to manage database access control within a Java application, according to three main directions: introduction of class diagrams towards Designer analysis and design models; use of Designer features to design user and security policy and generate database schema; reconcile object-oriented and relational worlds with TopLink and a dedicated security policy generator.

Work remains to be done to improve security policy patterns and to use stored procedures together with CRUD features.

7. Références

[1] Paul Dorsey and Peter Koletzke, Designer/2000 Handbook, Oracle Press, 1997.

[2] Richard Barker, Case Method: Tasks and Deliverables, Addison-Wesley Longman, 1990.

[3] Ivar Jacobson, Grady Booch, James Rumbaugh, The Unified Software Development Process, Addison-Wesley Longman, 1999.

[4] Eric J. Naiburg and Robert A. Maksimchuk, UML for Database design, Addison-Wesley Longman, 2001.

[5] ISO 10303-1. Part 1 : STEP overview and fundamentals principles, ISO, 1994.

[6] Alain Plantec and Vincent Ribaud. EUGENE: a STEP-based framework to build Application Generators. AWCSET'98, CSIRO-Macquarie University, 1998.

[7] Peter Koletzke, Paul Dorsey and Avrom Faderman, Oracle9i JDeveloper Handbook, Oracle Press, MCGraw-Hill, 2003.