Lightweight Transformation of Data Models from SQL Schemas to UML-ER

Technical Report 2009-565

Manar H. Alalfi James R. Cordy Thomas R. Dean

School of Computing, Queen's University, Kingston, Canada

{alalfi, cordy, dean}@cs.queensu.ca

Abstract

Data modeling is an essential part of the software development process, and together with application modeling forms the core of the model-driven approach to software engineering. While UML is considered the standard for application modeling, there is really no corresponding open standard for data modeling. In this paper, we propose an approach and a tool to help bridge the gap between application and data modeling based on source transformation technology. In this paper we introduce SQL2XMI, a tool that automatically transforms an SQL schema into a UML-ER model expressed in XML Meta Interchange (XMI) 2.1, which is specifically designed to support interchange of UML models. By bringing the data model to the UML world, our tool enables the manipulation and integration of both data and application models using the same UML-based tools in an interoperable way. While SQL2XMI was initially created to support our work in software comprehension, the approach we have used can be generalized to recover a rich UML-ER model from any SQL schema to any XMI 2.x format, and can easily be extended to support code engineering by automatically generating SQL schemas from an XMI 2.x file.

1 Introduction

Model-driven software development generally begins with either application modeling or data modeling. The two are closely related to, and complement, one another. In the application modeling domain, several Object Oriented modeling notations were combined in the early 1990s to produce the Unified Modeling Language (UML [22]). It has become the open

[1]E-mail: {alalfi, cordy, dean}@cs.queensu.ca

standard notation for describing multiple aspects of the specification and design of large software systems. On the other hand, Entity-Relationship diagrams (ER diagrams [8]) are usually used for data modeling, and ER is the most commonly used method to build data models for relational databases.

However, ER modeling does not define a standard graphical syntax for the representation of ER diagrams, and there is no open standard for representing data objects in ER. Rather, each practical modeling methodology uses its own notation. The original notation used by Chen [8] is widely used in academic texts and journals, but is rarely seen in either CASE tools or publications by non-academics. Currently, a number of notations used; among the more common are Bachman [5], crow's foot [27], and IDEFIX [13]. This diversity in the underlying notations leads to data modeling that is tightly coupled to the specific data modeling tool of a specific vendor. As a result there is neither a generally accepted open standard nor interoperability of ER data models. From a model-driven engineering (MDE) perspective there is no general way to define transformations that either generate or consume such data models.

As researchers try to find a standardized graphical representation for data modeling using ER diagrams, the obvious choice is to use UML. Using UML as a standard for ER data modeling could bring many benefits. First, because UML it is a widely accepted language used by analysts and software developers, it can be an excellent fit for graphical representation of ER diagrams. Second, UML's generality can assist in the unification of all areas of expertise into a unified platform. It does not matter how different the technologies are, all of them can be described using the same language. This strength is brought to UML by *profiles*, a standardized set of extensions and constraints that tailors UML to particular uses [23]. Finally, recent empirical research comparing ER with UML class diagrams indicates that UML can significantly improve the comprehension level of programmers [11].

Through UML, development teams for application modeling and data modeling can more easily communicate. In addition they can gain the benefits of easy integration into repositories using meta-models, use of a standardized input/output format (XMI), and a unified software development process from analysis to implementation to deployment. In the long run, other benefits related to code engineering may be gained, for example by using the UML model to generate Data Definition Language (DDL) scripts, which can be executed on database systems to create the specified tables. In the shorter term, reverse engineering of independently created SQL database definitions (DDL scripts) to UML models can yield some benefits for reasoning about and maintaining deployed production systems.

Unfortunately, up until now there is no standardized UML data modeling profile. The need for such profile has already forced some UML vendors and users to define their own UML profiles, but each has made their own interpretation and trade-offs, and all are UML 1.x based profiles.

This paper is an extended version of our previous short paper [2] in which we proposed a transformation technique to bridge the gap between data modeling and application modeling using UML. An automated transformation from an SQL (DDL) schema to an open XMI 2.1 UML-adapted class model is presented. The adapted model is a tailored UML class model

to represent the basic ER diagram components, including entities, attributes, relations, and primary keys. Our transformation technique is a novel one in that it is open, non-vendor specific, and targeted at the standard UML 2.1 exchange format, XMI 2.1. Although comparable commercial transformations exist, they are closed technologies targeted at formats tightly coupled to the vender's tools, hindering portability and preventing users from choosing their preferred tools in the development process.

While so far a prototype, our tool can recover a UML-based ER diagram from any SQL DDL schema and visualize it using any UML tool that supports the import of open standard XMI 2.1 exchange format. The tool can be easily adapted to be compatible with many different implementations of the SQL DDL notation, as well as different XMI 2.x versions. Using our method an open reverse transformation from XMI 2.x to an SQL DDL schema can also be easily implemented.

This paper is structured as follows. We begin in Section 2 by reviewing the related work. Section 3 introduces the SQL2XMI tool and briefly discusses its implementation using source transformations. Section 4 presents a small case study, using the tool to recover a UML ER diagram from the SQL DDL database schema of a popular open source production web application. Finally Section 5 concludes the paper and outlines possible directions for future work.

2 Previous Work

Several UML data modeling profiles (e.g., Ambler [3][4], Gorp [15], Gronik[14], and Silingas and Kaukenas [25]) have been proposed to answer the need for a UML data modeling profile to support entire development process and help integrate with application modeling. In the absence of a formal UML data modeling profile, the proposed profiles represent a partial solution at best, since they support only UML 1.x, do not address the interoperability issue, and do not support the need for valid transformations to generate and consume these models.

Many commercial tools, such as Rational Data Architect(RDA) [16], Rational Software Architect(RSA) [17] and MagicDraw [20] provide their users with a transformation facility to import SQL DDL schemas to ER diagrams. However, the transformations used in these systems involve specific proprietary file formats that are tied to their own tools, making portability difficult. For example, IBM tightly integrates RSA, which is used for application modeling, with RDA, used for data modeling. They have a transformation from an SQL DDL schema to an RDA logical data model (LDM) which can then be transformed into an RSA UML model and vice versa, but they do not provide a facility for direct transformation from an SQL DDL schema to an RSA UML model [7] and do not provide open interoperation. By contrast, our tool supports direct transformation from SQL DDL to the open standard XMI 2.1 portable interchange format for UML 2.1, thus supporting direct semantic model interchange between the range of different UML tools that are able to import the XMI 2.1 standard, including RSA. In addition, because our tool is open rather than proprietary, it can be quickly and easily adapted to support other XMI 2.x versions that may be used by other UML tools.

We use the TXL source transformation system [10] to implement our tool. This technology has previously been used

by Abu-Hamdeh et al. [1] to reverse engineer SQL schemas to Prolog-style textual factbases. While Abu-Hamdeh et al.'s transformation recovers more information than our tool, for example entity types (weak, strong, etc.), cardinalities and some constraints, our target representation is the XMI 2.1 model interchange standard, which can be easily imported, visualized and manipulated by UML-based tools. The additional information recovered by Abu-Hamdeh et al. can be handled by our tool in future using additional transformation rules, but is not required in our current program comprehension application.

Another related system is Chung and Hartford's XMI2SQL [9], which can transform an XMI file exported by a UML tool to an SQL implementation. An ER model based on the Ambler profile [3] is built in UML first, and then exported to XMI by the UML tool. XMI2SQL then transforms this Ambler-based model in XMI to an SQL DDL schema. XMI2SQL is implemented as a web service in C#. This tool complements our work on the forward engineering and code generation side. However, its adaptation to cope with different variations of XMI files and SQL schemas could be challenging, since it is implemented as a traditional custom C# program. Our use of source transformation rules makes it easy to rapidly adapt to variations.

There is a long history of reverse engineering of ER diagrams from databases, such as Premerlani and Blaha [24]. These approaches are more mature and handle more features by utilizing more input artifacts. Di Lucca et al. [12], Yang et al. [28] and Canfora et al. [6] all recover data models from the source code of data intensive applications. The purpose of our work is different, providing a lightweight translation of SQL DDL schemas to standard UML, a problem the OMG called for proposals for in 2005 [21].

3 SQL2XMI

SQL2XMI is a process for automatically transforming an SQL DDL database schema to a UML ER diagram in XMI 2.1 model exchange format. In this first implementation, we have targeted MySQL, although extending to any other SQL variants is straightforward.

The SQL grammar used in our project is tuned for MySQL version 3.23.44. MySQL supports multiple storage engines. The InnoDB engine supports the checking of foreign key constraints and enforcement of referential integrity. This support requires the explicit use of foreign key and reference attributes during table creation. Other engines ignore these attributes. When provided, SQL2XMI uses these attributes to recover relations between entities. Without the attributes, SQL2XMI recovers relations based on naming conventions. Identifying foreign key that are not explicitly defined is a problem that has been researched intensively in the past [24]. However, these approaches (e.g. [12, 28, 6]) require analysis of source code, which is difficult in dynamic dispatch systems such as PHP. Naming convention has proved sufficient for our purposes.

Our present prototype reverse engineers all the basic elements of the ER diagram, that is, the set of entities and their attributes, the primary key set, the foreign key set, and the relationships between them. In this initial version we do not infer entity types or cardinalities, although they can be easily inferred if required. Our initial application is the complex

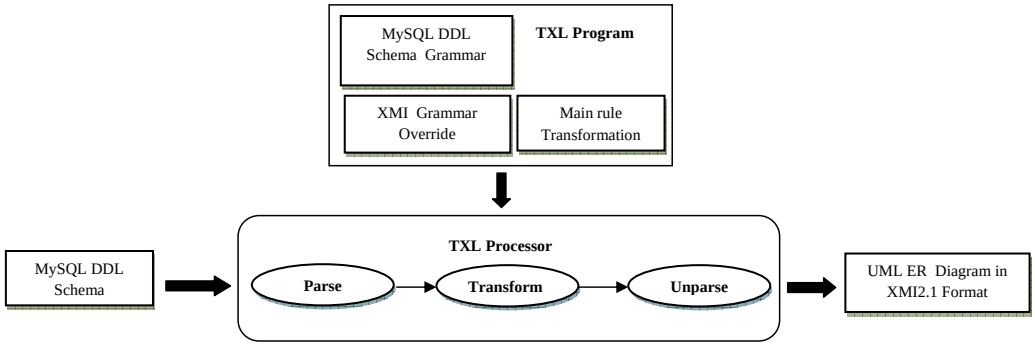

Figure 1. The TXL Transformation Technique as Applied in our Tool

software comprehension of web applications for security analysis, and in this application entity types and cardinalities are not a concern.

3.1 Implementation

Our implementation is based on source transformation technology, in which the target program, an XMI file, is viewed as a syntactic modification of the source program, the SQL DDL schema. This modification involves adding the target language features as extensions to the grammar of the source language, and then transforming each input language grammatical form to corresponding target language forms. For this purpose we use TXL [10], a programming language designed for manipulating and experimenting with programming language notations and features. TXL has been used in many production applications with transformations involving billions of lines of source code.

The TXL transformation process normally consists of three parts: a context-free "base" grammar for the source language to be manipulated, a set of context-free grammatical "overrides" (extensions or changes) to the base grammar, and a rooted set of source transformation rules to implement transformation of the extensions to the base language [10]. The TXL processer parses the source program and converts it into a parse tree, then recursively applies the set of transformation rules, beginning with a "main" rule, until there is no match encountered in the parse tree. TXL finalizes the process by unparsing the transformed parse tree into the target program. Figure 1 shows the TXL transformation process as applied in our tool. In our application we began by defining a grammar for the input MySQL DDL, and a second (override) grammar for our target XMI output forms. The main grammatical form (called [program] in TXL) allows both, but expects the XMI part to be empty on input and full on output, and vice-versa for the SQL part 2. We used the TXL producer-consumer translation paradigm to make a set of transformation rules to "produce" the XMI output while "consuming" the SQL input. The transformation is started by the main rule, shown in Figure 2. TXL begins by parsing the MySQL input schema into its basic DDL statements (pattern variable *MySQLs*), with an initially empty XMI output (pattern variable *XMIFile*). The main rule replaces this entire input by constructing the XMI output beginning with the empty *XMIFile* and transforming it into the XMI representation of the input *MySQLs* using three transformation subrules, *createXMIH*, *GenerateERDElements*, and *createXMIE*.

```
% Base grammar for SQL
include "MySQLGrammar.grm"

% Output override grammar for XMI
include "XMI.grm"

% Allow for both input and output forms
define program
   [repeat MySQLStatement]
   [repeat XMItoken]
end define

% Main transformation rule
function main
   replace [program]
      MySQLS  [repeat MySQLStatement]
      XMIFile [repeat XMItoken]
   by
      XMIFile            % Initially empty
         [createXMIH]    % Make XMI header
         [GenerateERDElements MySQLS each MySQLS]
                         % Generate XMI elements
         [createXMIE]    % Make XMI trailer
end function
```

Figure 2. Grammatical specification and main transformation rule (function)

The functions *createXMIH* and *createXMIE* simply create the XMI header and trailer elements which bracket the body of the translation. The function *GenerateERDElements*, shown in Figure 3, is responsible for the bulk of the transformation, transforming each SQL table definition, including all of its attributes, relations, and primary keys, to the corresponding XMI 2.1 elements. The correspondence between SQL DDL schema elements and XMI 2.1 elements (i.e. UML 2.1 ER diagram elements) implemented by the transformation is shown in Table 4. The *GenerateERDElements* function uses pattern matching to recognize each of the DDL table elements and map them to the corresponding XMI output.

3.2 Transforming Table Definitions to Entities, Attributes and Relationships

For the transformation of each SQL table in the input, the *GenerateERDElements* function is passed both the particular table to be transformed (**each** `MySQLs`) and the entire set of all of the input tables (`MySQLs`). This is because SQL tables do not have a simple one-to-one relationship with the corresponding UML 2.1 ER model - rather, the transformation of each table definition depends on information that can only be derived from other tables to which it is related, for example the set of all primary keys in the schema. Thus it is a context-dependent source transformation.

The function begins by deconstructing the `create table` statement to break it into its syntactic parts so we can access and manipulate them separately. By doing so, we are able to extract the table name to map to the entity name, the table columns to map to the entity attributes, and to identify the table's primary keys and relations shared between the entities. Our implementation involves 16 separate transformation rules. Here we discuss three of the critical transformations in detail, and briefly outline the overall transformation process. Other rules in our transformation are similar.

The TXL constructor *SetOfAllPK* collects a list of all *(primary key, table name)* pairs based on the explicit primary key constraint statements in all of the table definitions in the database. This information is required later in the rule for inferring

```
% Generate the UML ER diagram XMI element for each table column
function GenerateERDElements
            AllTableStructures [repeat MySQLStatement]
            TableStructure [MySQLStatement]
   deconstruct TableStructure
      CREATE TABLE TN [id] ( ColList [list createDefinition] );
   construct Tname [stringlit]
      _ [quote TN]
   construct SetOfAllPK [repeat XMItoken]
      _ [findAllPKL each AllTableStructures]
   construct SetOfPK [list index_col_name]
      _ [findPKL each ColList]
   construct eAnnotationsID [stringlit]
      _ [quote TN ] [+ "EAnnotation" ]
   construct detailsID [stringlit]
      _ [quote TN ] [+ "_Entity" ]
   construct XMI_ERD_Entity [repeat XMItoken]
      <packagedElement
         xmi:type="uml:Class" xmi:id=Tname name=Tname>
      <xmi:Extension
         extender="http://www.eclipse.org/emf/2002/Ecore">
         <eAnnotations xmi:type="ecore:EAnnotation"
               xmi:id= eAnnotationsID
               source="http://www.eclipse.org/uml2/2.0.0/UML">
            <details xmi:type="ecore:EStringToStringMapEntry"
               xmi:id= detailsID key="Entity"/>
         </eAnnotations>
      </xmi:Extension>
   construct packagedElementCloseT [repeat XMItoken]
      </packagedElement>
   construct XMI_ERD [repeat XMItoken]
      XMI_ERD_Entity
         [createEntityAttrib Tname SetOfAllPK each ColList]
         [constructFKside_Relation TableStructure each SetOfAllPK]
         [constructRelations TN  AllTableStructures each SetOfPK]
         [. packagedElementCloseT]
         [constructAss TN  SetOfPK each AllTableStructures]
   replace * [repeat XMItoken]
     % tail of output
   by
      XMI_ERD
end function
```

Figure 3. The GenerateERDElements function, which transforms each SQL table to its representation in XMI

Relational Database Element	Entity Relation Diagram Element	UML Class Diagram XMI elements (tailored to ER diagram)						
		XMI Tag	XMI Type	XMI ID	XMI Name	XMI Extension or additional attributes		
Table	<<Entity>>	packagedElement	uml:Class	Table name	Table name	eAnnotations extension with a detail element	type	ecore: EStringToStringMapEntry
							id	Unique ID
							Key	"Entity"
Column	Attribute	ownedAttribute	uml:Property	Attribute Name	Attribute Name	Additional attributes	visibility	private|public|..
							type	Attribute Data Type
Primary Key	Attribute With extension (stereotype annotation) <<PK>>	ownedAttribute	uml:Property	Attribute Name	Attribute Name	Additional attributes	visibility	private|public|..
							type	Attribute Data Type
						eAnnotations extension with a detail element	type	ecore: EStringToStringMapEntry
							id	Unique ID
							Key	"PK"
Foreign Key	Attribute	ownedAttribute	uml:Property	Source Table Name_ Target Table Name	Target Table Name(PK)	Additional attributes	visibility	private|public|..
							type	Target Table Name
							association	AssociationID
Association	Relation	packagedElement	uml:Association	AssociationID	NULL	Additional attributes	memberEnd	String type containing the id of entities member in this relation

Figure 4. Mappings between MySql schema elements, ERD elements, and XMI 2.1 elements

entity relationships for the table being processed. The constructor *SetOfPK* then collects another list of all the primary keys defined in the particular table being processed.

The constructor XMI_ERD_Entity creates the XMI representation of the entity for the table, consisting of an XMI *packagedElement* element of type *uml:Class* that is annotated to be an *Entity* using stereotyping. The constructor XMI_ERD then uses a number of subrules to flesh out this initial ER representation of the table by adding the translation of attributes, foreign keys and relationships to yield the entire translation of the SQL table definition.

3.3 Transforming Table Columns to Attributes

For each table column, the *GenerateERDElements* function uses the transformation subrule *createEntityAttribute*, shown in Figure 5. This transformation function generates an XMI *ownedAttribute* of type *uml:Property* to represent the table column. It begins by using TXL deconstructors pattern to capture the column's name(*ColName*), definition (*ColDef*), and data type (*DT*). The TXL constructor uses the transformation subrule *IsPKAttrib* to determine if the column is a primary key of the table and to annotate it with the XMI primary key stereotype if so. The remainder of the function creates the *ownedAttribute* representation of the column in XMI, embeds the primary key stereotyping and appends the result to the XMI entity representation of the table.

The transformation subrule *IsPKAttrib*, shown in Figure 6, identifies table primary keys by checking if the table column *ColN* and its table *STableName* are present in the set of all primary keys that was collected in *GenerateERDElements* and passed as parameter to its subrules. If so, an XMI primary key stereotype *eAnnotation* element that corresponds to the matched table column is returned by the function (otherwise it returns no annotation). The annotation is added to the XMI representation of the column by *GenerateERDElements* in its translation.

```
% Generate the UML ER diagram XMI element
% for each table column

function createEntityAttrib  STableName [stringlit]
      PKL [ repeat XMItoken] Colm [createDefinition]

   deconstruct Colm
      ColName [colm_name] ColDef [col_def]

   deconstruct * [dataType] ColDef
      DT [dataType

   % isPKAttrib checks if the column is primary key
   % and annotates it with PK stereotype if so
   construct ModefidColName [repeat XMItoken]
      _ [isPKAttrib STableName ColName each PKL]

   construct AttribName [stringlit]
      _ [quote ColName]

   construct AttribDT [stringlit]
      _ [quote DT]

   construct AttribDef [repeat XMItoken]
      <ownedAttribute xmi:type="uml:Property"
         xmi:id= AttribName name= AttribName
         visibility="private">

   construct Closingtag [repeat XMItoken]
      </ownedAttribute>

   replace * [repeat XMItoken]
      % tail of output
   by
      AttribDef [.ModefidColName] [.Closingtag]
end function
```

Figure 5. The createEntityAttribute function, which translates SQL table columns to attributes in XMI

```
% Annotate XMI attribute element with PK stereotype
% if primary key

function IsPKAttrib STableName [stringlit]
            ColN [colm_name] SetOfAllPK [XMItoken]

   construct ColN_string [stringlit]
      _ [ quote ColN]

   % Is this column of this table in the primary keys?
   deconstruct SetOfAllPK
      <ownedAttribute _ [opt xmi_colon]
            _ [id_or_key] = STableName]
            _ [id_or_key] = ColN_string >

   replace * [repeat XMItoken]
      % tail of output
   by
      <xmi:Extension
          extender="http://www.eclipse.org/emf/2002/Ecore">
        <eAnnotations xmi:type="ecore:EAnnotation"
            xmi:id="_Ovi-uPdEdy8F"
            source="http://www.eclipse.org /uml2/2.0.0/UML">
          <details
            xmi:type="ecore:EStringToStringMapEntry"
            xmi:id="_Ovi-V- 8rGg0Zw" key="PK"/>
        </eAnnotations>
      </xmi:Extension>
end  function
```

Figure 6. The IsPKAttribute function, which identifies and stereotypes the XMI representation of columns which are primary keys

3.4 Transforming Foreign Keys and Relations to Relationships and Associations

When foreign key and references attributes exist in the schema, they can be used by SQL2XMI to identify relations. When absent, SQL2XMI uses naming convention and column data type to infer foreign key relations. This can be augmented in the future with stemming operations to enhance naming conventions. Otherwise, foreign key inference is a difficult problem that requires additional data and code analysis such as the work by Di Lucca et al. [12], Yang et al. [28] and Canfora et al. [6], which requires a completed running application and is outside the scope of our work. The following are the steps used to recover relations between tables based on naming convention and the data type similarity, as the other case is straightforward:

First, following the transformation of table columns, the *GenerateERDElements* function uses similar transformation subrules to handle foreign keys and relations. The subrule *constructFKside_Relation* identifies foreign keys as relation target by checking for an occurrence of each KEY column in the set of all primary keys *SetOfAllPK*, which is passed as a parameter to the subrule. For each primary key of another table T_i that matches, an XMI *ownedAttribute* element of type *uml:Property* to refer to a relation of type T_i between the two tables is generated, where T_i is the name of the other table.

Next, *GenerateERDElements* uses the transformation subrule *constructRelations* to identify foreign keys as relation source by checking for columns that are primary keys in this table and also occur as a KEY column in another table. For each such foreign key, an XMI *ownedAttribute* element of type *uml:Property* is generated to refer to a relation of type T_i between the two tables.

Finally, *GenerateERDElements* uses the transformation subrule *constructAss* to create an XMI *packagedElement* of type *uml:Association* between the two tables involved in each relation generated by the previous two subrules.

The concatenation of the results of the entire set of transformation subrules of *GenerateERDElements* forms the complete result transformation of the column to XMI, and the concatenation of the transformations of the whole set of columns forms the result of the main transformation function, yielding the complete XMI 2.1 representation of the UML 2.1 ER diagram for the original SQL schema.

4 An Example: PhpBB

SQL2XMI was originally designed to serve an ongoing project in web application security analysis, in which reverse engineering based on both static and dynamic analysis is being used to identify the application resources, permissions, and subjects that constitute the basic elements of a security system. Data models constitute one of the main sources of such information, and visualizing data models facilitates the process of understanding the structure of the system, its basic entities and their relationships. The output of the understanding phase is mainly represented using UML models, and we are using the XMI 2.1 model interchange notation to help unify the results of the understanding phases of our project.

In this context, we have evaluated SQL2XMI on the popular web bulletin board system PhpBB versions 2.0 and 3.0. Using

```
CREATE TABLE phpbb_forums (
   forum_id smallint(5) UNSIGNED NOT NULL,
   cat_id mediumint(8) UNSIGNED NOT NULL,
   forum_name varchar(150),
   forum_desc text,
   forum_status tinyint(4) DEFAULT '0' NOT NULL,
   forum_order mediumint(8) UNSIGNED DEFAULT '1' NOT NULL,
   forum_posts mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   forum_topics mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   forum_last_post_id mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   prune_next int(11),
   prune_enable tinyint(1) DEFAULT '0' NOT NULL,
   auth_view tinyint(2) DEFAULT '0' NOT NULL,
   auth_read tinyint(2) DEFAULT '0' NOT NULL,
   auth_post tinyint(2) DEFAULT '0' NOT NULL,
   auth_reply tinyint(2) DEFAULT '0' NOT NULL,
   auth_edit tinyint(2) DEFAULT '0' NOT NULL,
   auth_delete tinyint(2) DEFAULT '0' NOT NULL,
   auth_sticky tinyint(2) DEFAULT '0' NOT NULL,
   auth_announce tinyint(2) DEFAULT '0' NOT NULL,
   auth_vote tinyint(2) DEFAULT '0' NOT NULL,
   auth_pollcreate tinyint(2) DEFAULT '0' NOT NULL,
   auth_attachments tinyint(2) DEFAULT '0' NOT NULL,
   PRIMARY KEY (forum_id),
   KEY forums_order (forum_order),
   KEY cat_id (cat_id),
   KEY forum_last_post_id (forum_last_post_id));

CREATE TABLE phpbb_forum_prune (
   prune_id mediumint(8) UNSIGNED NOT NULL auto_increment,
   forum_id smallint(5) UNSIGNED NOT NULL,
   prune_days smallint(5) UNSIGNED NOT NULL,
   prune_freq smallint(5) UNSIGNED NOT NULL,
   PRIMARY KEY(prune_id),
   KEY forum_id (forum_id));

CREATE TABLE phpbb_categories (
   cat_id mediumint(8) UNSIGNED NOT NULL auto_increment,
   cat_title varchar(100),
   cat_order mediumint(8) UNSIGNED NOT NULL,
   PRIMARY KEY (cat_id),
   KEY cat_order (cat_order));

CREATE TABLE phpbb_user_group (
   group_id mediumint(8) DEFAULT '0' NOT NULL,
   user_id mediumint(8) DEFAULT '0' NOT NULL,
   user_pending tinyint(1),
   PRIMARY KEY (group_id, user_id),
   KEY group_id (group_id),
   KEY user_id (user_id));

CREATE TABLE phpbb_posts (
   post_id mediumint(8) UNSIGNED NOT NULL auto_increment,
   topic_id mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   forum_id smallint(5) UNSIGNED DEFAULT '0' NOT NULL,
   poster_id mediumint(8) DEFAULT '0' NOT NULL,
   post_time int(11) DEFAULT '0' NOT NULL,
   poster_ip char(8) NOT NULL,
   post_username varchar(25),
   enable_bbcode tinyint(1) DEFAULT '1' NOT NULL,
   enable_html tinyint(1) DEFAULT '0' NOT NULL,
   enable_smilies tinyint(1) DEFAULT '1' NOT NULL,
   enable_sig tinyint(1) DEFAULT '1' NOT NULL,
   post_edit_time int(11),
   post_edit_count smallint(5) UNSIGNED DEFAULT '0' NOT NULL,
   PRIMARY KEY (post_id),
   KEY forum_id (forum_id),
   KEY topic_id (topic_id),
   KEY poster_id (poster_id),
   KEY post_time (post_time));


CREATE TABLE phpbb_posts_text (
   post_id mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   bbcode_uid char(10) DEFAULT '' NOT NULL,
   post_subject char(60),
   post_text text,
   PRIMARY KEY (post_id) );


CREATE TABLE phpbb_auth_access (
   group_id mediumint(8) DEFAULT '0' NOT NULL,
   forum_id smallint(5) UNSIGNED DEFAULT '0' NOT NULL,
   auth_view tinyint(1) DEFAULT '0' NOT NULL,
   auth_read tinyint(1) DEFAULT '0' NOT NULL,
   auth_post tinyint(1) DEFAULT '0' NOT NULL,
   auth_reply tinyint(1) DEFAULT '0' NOT NULL,
   auth_edit tinyint(1) DEFAULT '0' NOT NULL,
   auth_delete tinyint(1) DEFAULT '0' NOT NULL,
   auth_sticky tinyint(1) DEFAULT '0' NOT NULL,
   auth_announce tinyint(1) DEFAULT '0' NOT NULL,
   auth_vote tinyint(1) DEFAULT '0' NOT NULL,
   auth_pollcreate tinyint(1) DEFAULT '0' NOT NULL,
   auth_attachments tinyint(1) DEFAULT '0' NOT NULL,
   auth_mod tinyint(1) DEFAULT '0' NOT NULL,
   PRIMARY KEY (group_id, forum_id),
   KEY group_id (group_id),
   KEY forum_id (forum_id));
```

```
CREATE TABLE phpbb_users (
   user_id mediumint(8) NOT NULL,
   user_active tinyint(1) DEFAULT '1',
   username varchar(25) NOT NULL,
   user_password varchar(32) NOT NULL,
   user_session_time int(11) DEFAULT '0' NOT NULL,
   user_session_page smallint(5) DEFAULT '0' NOT NULL,
   user_lastvisit int(11) DEFAULT '0' NOT NULL,
   user_regdate int(11) DEFAULT '0' NOT NULL,
   user_level tinyint(4) DEFAULT '0',
   user_posts mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   user_timezone decimal(5,2) DEFAULT '0' NOT NULL,
   user_style tinyint(4),
   user_lang varchar(255),
   user_dateformat varchar(14) DEFAULT 'd M Y H:i' NOT NULL,
   user_new_privmsg smallint(5) UNSIGNED DEFAULT '0' NOT NULL,
   user_unread_privmsg smallint(5) UNSIGNED DEFAULT '0' NOT NULL,
   user_last_privmsg int(11) DEFAULT '0' NOT NULL,
   user_login_tries smallint(5) UNSIGNED DEFAULT '0' NOT NULL,
   user_last_login_try int(11) DEFAULT '0' NOT NULL,
   user_emailtime int(11),
   user_viewemail tinyint(1),
   user_attachsig tinyint(1),
   user_allowhtml tinyint(1) DEFAULT '1',
   user_allowbbcode tinyint(1) DEFAULT '1',
   user_allowsmile tinyint(1) DEFAULT '1',
   user_allowavatar tinyint(1) DEFAULT '1' NOT NULL,
   user_allow_pm tinyint(1) DEFAULT '1' NOT NULL,
   user_allow_viewonline tinyint(1) DEFAULT '1' NOT NULL,
   user_notify tinyint(1) DEFAULT '1' NOT NULL,
   user_notify_pm tinyint(1) DEFAULT '0' NOT NULL,
   user_popup_pm tinyint(1) DEFAULT '0' NOT NULL,
   user_rank int(11) DEFAULT '0',
   user_avatar varchar(100),
   user_avatar_type tinyint(4) DEFAULT '0' NOT NULL,
   user_email varchar(255),
   user_icq varchar(15),
   user_website varchar(100),
   user_from varchar(100),
   user_sig text,
   user_sig_bbcode_uid char(10),
   user_aim varchar(255),
   user_yim varchar(255),
   user_msnm varchar(255),
   user_occ varchar(100),
   user_interests varchar(255),
   user_actkey varchar(32),
   user_newpasswd varchar(32),
   PRIMARY KEY (user_id),
   KEY user_session_time (user_session_time));

CREATE TABLE phpbb_topics (
   topic_id mediumint(8) UNSIGNED NOT NULL auto_increment,
   forum_id smallint(8) UNSIGNED DEFAULT '0' NOT NULL,
   topic_title char(60) NOT NULL,
   topic_poster mediumint(8) DEFAULT '0' NOT NULL,
   topic_time int(11) DEFAULT '0' NOT NULL,
   topic_views mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   topic_replies mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   topic_status tinyint(3) DEFAULT '0' NOT NULL,
   topic_vote tinyint(1) DEFAULT '0' NOT NULL,
   topic_type tinyint(3) DEFAULT '0' NOT NULL,
   topic_first_post_id mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   topic_last_post_id mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   topic_moved_id mediumint(8) UNSIGNED DEFAULT '0' NOT NULL,
   PRIMARY KEY (topic_id),
   KEY forum_id (forum_id),
   KEY topic_moved_id (topic_moved_id),
   KEY topic_status (topic_status),
   KEY topic_type (topic_type));

CREATE TABLE phpbb_topics_watch (
  topic_id mediumint(8) UNSIGNED NOT NULL DEFAULT '0',
  user_id mediumint(8) NOT NULL DEFAULT '0',
  notify_status tinyint(1) NOT NULL default '0',
  PRIMARY KEY (topic_id,user_id,notify_status),
  KEY topic_id (topic_id),
  KEY user_id (user_id),
  KEY notify_status (notify_status));

CREATE TABLE phpbb_groups (
   group_id mediumint(8) NOT NULL auto_increment,
   group_type tinyint(4) DEFAULT '1' NOT NULL,
   group_name varchar(40) NOT NULL,
   group_description varchar(255) NOT NULL,
   group_moderator mediumint(8) DEFAULT '0' NOT NULL,
   group_single_user tinyint(1) DEFAULT '1' NOT NULL,
   PRIMARY KEY (group_id),
   KEY group_single_user (group_single_user));
```

Figure 7. A part of the PhpBB 2.0 MySql schema

```
<uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ecore="
http://www.eclipse.org/emf/2002/Ecore" xmlns:uml="http://schema.omg.org/spec/UML/2.1.1" xsi:schemaLocation="http://schema.omg.org/spec/UML/2.1.1
http://www.eclipse.org/uml2/2.0.0/UML" xmi:id="_XrSbENk5EdyEZoPpUv3LUw" name="Blank Model">
 <packageImport xmi:type="uml:PackageImport" xmi:id="_XrSbEdk5EdyEZoPpUv3LUw">
     <importedPackage xmi:type="uml:Model" href="http://schema.omg.org/spec/UML/2.1.1/uml.xml#_0"/>
 </packageImport>
 <packagedElement xmi:type="uml:Class" xmi:id="phpbb_groups" name="phpbb_groups">
     <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
         <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="phpbb_groupsEAnnotation" source="http://www.eclipse.org/uml2/2.0.0/UML">
             <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="phpbb_groups_Entity" key="Entity"/>
         </eAnnotations>
     </xmi:Extension>
     <ownedAttribute xmi:type="uml:Property" xmi:id="group_id" name="group_id" visibility="private">
         <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
             <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Ovi-VuPdEdy8F_b8rGg0Zw" source="http://www.eclipse.org/uml2/2.0.0/UML">
                 <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Ovi-V-PdEdy8F_b8rGg0Zw" key="PK"/>
             </eAnnotations>
         </xmi:Extension>
     </ownedAttribute>
     <ownedAttribute xmi:type="uml:Property" xmi:id="group_type" name="group_type" visibility="private">
     </ownedAttribute>
     <ownedAttribute xmi:type="uml:Property" xmi:id="group_name" name="group_name" visibility="private">
     </ownedAttribute>
     <ownedAttribute xmi:type="uml:Property" xmi:id="group_description" name="group_description" visibility="private">
     </ownedAttribute>
     <ownedAttribute xmi:type="uml:Property" xmi:id="group_moderator" name="group_moderator" visibility="private">
     </ownedAttribute>
     <ownedAttribute xmi:type="uml:Property" xmi:id="group_single_user" name="group_single_user" visibility="private">
     </ownedAttribute>
     <ownedAttribute xmi:type="uml:Property" xmi:id="phpbb_groupsphpbb_auth_access_ass" name="phpbb_auth_access(group_id)" visibility="private"
     type="phpbb_auth_access" association="phpbb_auth_access__phpbb_groups__group_id"/>
     <ownedAttribute xmi:type="uml:Property" xmi:id="phpbb_groupsphpbb_user_group_ass" name="phpbb_user_group(group_id)" visibility="private"
     type="phpbb_user_group" association="phpbb_user_group__phpbb_groups__group_id"/>
     <ownedAttribute xmi:type="uml:Property" xmi:id="phpbb_groupsphpbb_auth_access_ass" name="phpbb_auth_access(group_id)" visibility="private"
     type="phpbb_auth_access" association="phpbb_groups__phpbb_auth_access__group_id"/>
     <ownedAttribute xmi:type="uml:Property" xmi:id="phpbb_groupsphpbb_user_group_ass" name="phpbb_user_group(group_id)" visibility="private"
     type="phpbb_user_group" association="phpbb_groups__phpbb_user_group__group_id"/>
 </packagedElement>
 <packagedElement xmi:type="uml:Association" xmi:id="phpbb_groups__phpbb_auth_access__group_id" memberEnd="phpbb_groupsphpbb_auth_access_ass
 phpbb_auth_accessphpbb_groups_ass"/>
 <packagedElement xmi:type="uml:Association" xmi:id="phpbb_groups__phpbb_user_group__group_id" memberEnd="phpbb_groupsphpbb_user_group_ass
 phpbb_user_groupphpbb_groups_ass"/>
 <packagedElement xmi:type="uml:Class" xmi:id="phpbb_auth_access" name="phpbb_auth_access">
         <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
             <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="phpbb_auth_accessEAnnotation" source="http://www.eclipse.org/uml2/2.0.0/UML">
                 <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="phpbb_auth_access_Entity" key="Entity"/>
             </eAnnotations>
         </xmi:Extension>
         <ownedAttribute xmi:type="uml:Property" xmi:id="group_id" name="group_id" visibility="private">
             <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
                 <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Ovi-VuPdEdy8F_b8rGg0Zw" source="http://www.eclipse.org/uml2/2.0.0/UML">
                     <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Ovi-V-PdEdy8F_b8rGg0Zw" key="PK"/>
                 </eAnnotations>
             </xmi:Extension>
         </ownedAttribute>
         <ownedAttribute xmi:type="uml:Property" xmi:id="forum_id" name="forum_id" visibility="private">
             <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
                 <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Ovi-VuPdEdy8F_b8rGg0Zw" source="http://www.eclipse.org/uml2/2.0.0/UML">
                     <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Ovi-V-PdEdy8F_b8rGg0Zw" key="PK"/>
                 </eAnnotations>
             </xmi:Extension>
         </ownedAttribute>
         <ownedAttribute xmi:type="uml:Property" xmi:id="auth_view" name="auth_view" visibility="private">
         </ownedAttribute>
         <ownedAttribute xmi:type="uml:Property" xmi:id="auth_read" name="auth_read" visibility="private">
         </ownedAttribute>
        .  .  .
    </packagedElement>
  . . . .
 <profileApplication xmi:type="uml:ProfileApplication" xmi:id="_XrSbHdk5EdyEZoPpUv3LUw">
    <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
        <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_XrSbHtk5EdyEZoPpUv3LUw" source="http://www.eclipse.org/uml2/2.0.0/UML">
           <references xmi:type="ecore:EPackage" href="http://schema.omg.org/spec/UML/2.1.1/StandardProfileL2.xmi#_yzU58YinEdqtvbnfB2L_5w"/>
        </eAnnotations>
    </xmi:Extension>
 <appliedProfile xmi:type="uml:Profile" href="http://schema.omg.org/spec/UML/2.1.1/StandardProfileL2.xmi#_0"/>
     </profileApplication>
/uml:Model>
```

Figure 8. Sample of the generated XMI 2.1 file

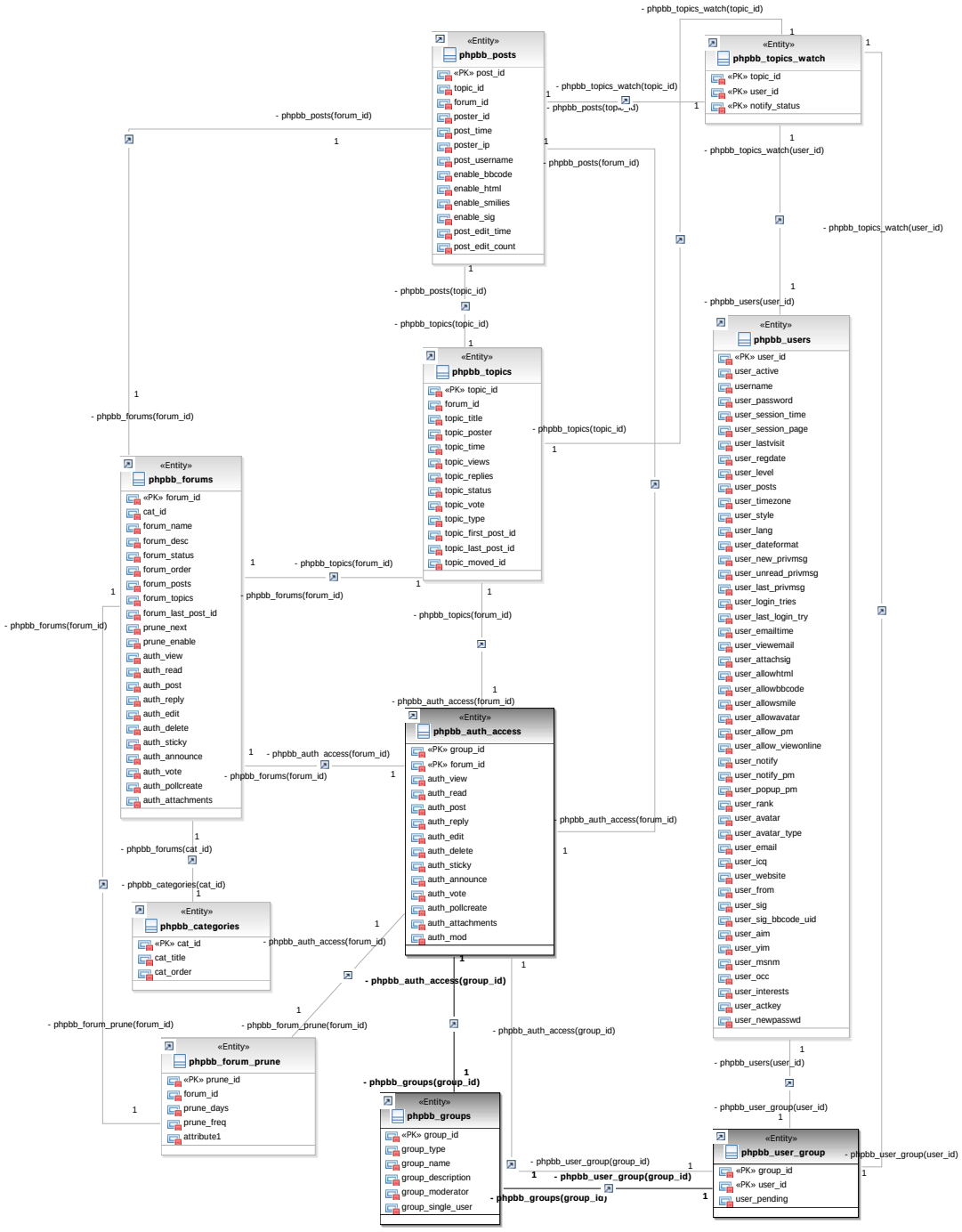

Figure 9. RSA visualization for a part of the generated diagram

the source transformation system described in Section 3.1, SQL2XMI has been able to automatically recover ER diagrams for the data models of both versions. The result has helped us both to understand the complex data model of this system and to recognize that the data model of PhpBB 3.0 has been completely restructured from the previous version.

PhpBB 2.0 has 30 tables that generate a 30 page XMI file representing a UML2 ER diagram much too large to fit in this paper. So as an example we show only a part of the schema including ten tables in Figure 7, one page of the generated XMI file, showing one entity and all of its relations in Figure 8, and a snapshot of part of the visualization of the ER diagram for the ten tables in Figure 9. We used Rational Software Architect (RSA) [17] to import and visualize the UML 2.1 ER diagram represented by the XMI 2.1 file automatically generated by our tool.

To get a better feeling for the process, we can look at a table from the schema presented in Figure 7 and see how it is mapped into the elements of the XMI file in Figure 8, and on to the ER diagram in Figure 9.

The table $phpbb\_groups$, which is the last table defined in the schema of Figure 7(A) has one primary key, $group\_id$, and six columns. The table does not define a foreign key statement, so it hard to tell from a first look at the schema which other tables are involved in relations with this table. In Figure 8, we can see that, based on the mappings defined on Table 4, the $phpbb\_groups$ table has been mapped into a UML *packagedElement* of type *"uml:Class"* and annotated as an *"Entity"* using the XMI tag *"eAnnotation"*.

The primary key, $group\_id$, has been mapped to an XMI *"ownedAttribute"* element with type *uml:property*, and annotated with the XMI *"eAnnotation"* element to be of type *"PK"*. Each of the other five columns is mapped to an XMI *"ownedAttribute"* element of type *"uml:property"*, but without any annotation. In the lower middle part of Figure 9 we can see the visualization of the XMI element and how it is represented as $<< Entity >>$, with the primary key represented as $<< PK >> group\_id$, and the other five attributes listed without any annotation.

We can recognize easily in Figure 9 that the entity $phpbb\_groups$, marked with (A), is involved in tow relations based on its primary key, one with $phpbb\_auth\_access$ entity, marked with (B), while the other with $phpbb\_user\_group$, marked with (C) . This fact is difficult to see in the original schema because the foreign key constraint is not defined explicitly. Relations for the table have been recovered as described in section II, and constructed in the XMI file as a *packageElement* of type *"uml:Association"* whose *memberEnds* are the entities evolved in the relation. We can recognize two of these in the XMI file, reflecting the two relations that the $phpbb\_groups$ entity is involved in. XMI *ownedAttribute* elements of XMI type *"uml:Property"* has been recovered for both entities that share a relation, such that each one refers to the other in the *type* attribute of the element, and both of them refer to the same *association* element that joins them.

Even in this first simple example we can see that important information such as relations can often not be easily understood directly from the SQL schema. The process of comprehending the application at this level can be tedious work, especially for more complicated and larger schemas. Similarly, even after automatic transformation to ER form, the XMI file structure is itself much too complicated to follow. Visualizing the XMI file as an ER diagram as shown in Figure 9, however, presents

the database schema in format that can be easily and quickly understood by all members of the development team.

5 Conclusions and Future Work

In this paper we have presented a source transformation technique to bridge the gap between data modeling and application modeling that can assist in the process of complex software comprehension and evolution. Our new open tool, SQL2XMI, automatically transforms an SQL DDL schema to a UML 2.1 ER diagram which can be visualized by any UML tool that supports XMI 2.1. Unlike other tools that reverse engineer to proprietary formats, SQL2XMI explicitly aims at open and flexible portability, requiring only the SQL DDL schema and targeting the official OMG XMI 2.1 UML representation. We have presented the details of our lightweight source transformation-based approach and an example of the application of our tool to recover an ER diagram for the popular internet bulletin board system PhpBB. The approach and mapping are unique to our work.

To bring our prototype tool to an industrial level, several improvements will be needed. It must be generalized to handle SQL database schemas other than MySQL and XMI 2.x versions other than 2.1. Using TXL gives us the ability to integrate handling of other implementations of the SQL standard quickly, simply by overriding the SQL grammar to add the forms of each vendor's specific extensions.

In this paper we have begun with just the MySql implementation of the SQL data definition language (DDL), leaving the improvement of the grammar file to include the data manipulation part (DML) and support for other vendors' implementations to future work. Our transformation also does not yet take advantage of all of the information available in the schema. Using a more comprehensive transformation rule set, we hope to recover a richer ER model.

Finally, while in this work we have concentrated on reverse engineering an existing MySql schema to a UML entity relationship diagram, in future we could use the same technique in the forward engineering direction, using the same technology to generate different SQL database implementations from an ER diagram designed using any UML toolset that supports XMI 2.1 export.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] Rateb Abu-Hamdeh, James R. Cordy, and T. Patrick Martin. Schema translation using structural transformation. In *CASCON*, pages 202–215, 1994.

[2] Manar H. Alalfi, James R. Cordy, and Thomas R. Dean. Sql2xmi: Reverse engineering of uml-er diagrams from relational database schemas. In *15th Working Conference on Reverse Engineering (WCRE 2008), Antwerp, Belgium, October*, pages 187–191, 2008.

[3] Scott Ambler. A UML profile for data modeling. www.agiledata.org: Techniques for Successful Evolutionary/Agile Database Development, 2006.

[4] Scott Ambler. *Agile database techniques*. John Wiley and Sons, Indianapolis, Indiana, USA, October 2003.

[5] Charles W. Bachman. Data structure diagrams. *SIGMIS Database*, 1(2):4–10, 1969.

[6] Gerardo Canfora, Aniello Cimitile, Andrea De Lucia, and Giuseppe A. Di Lucca. Decomposing legacy systems into objects: an eclectic approach. *Inf. & Soft. Tech.*, 43(6):401–412, 2001.

[7] Daniel T. Chang. Integrating Rational Software Architect with Rational Data Architect. IBM developerWorks, 2007.

[8] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Trans. Datab. Syst.*, 1(1):9–36, 1976.

[9] Sam Chung and Eric Hartford. Bridging the gap between data models and implementations: XMI2SQL. In *AICT/ICIW*, page 201, 2006.

[10] James R. Cordy. The TXL source transformation language. *Sci. Comput. Program.*, 61(3):190–210, 2006.

[11] Andrea De Lucia, Carmine Gravino, Rocco Oliveto, and Genoveffa Tortora. Data Model Comprehension: An Empirical Comparison of ER and UML Class Diagrams. In *ICPC*, pages 93–102, June 2008.

[12] Giuseppe A. Di Lucca, Anna Rita Fasolino, and Ugo de Carlini. Recovering class diagrams from data-intensive legacy systems. In *ICSM*, pages 52–63, 2000.

[13] Federal Information Processing Standards. Publication 184, Integration Definition for Information Modeling (IDEFIX), *http://www.itl.nist.gov/fipspubs/idef1x.doc*.

[14] Davor Gornik. UML data modeling profile. Technical report, IBM Rational Software Whitepaper TP 162 05/02, 2003.

[15] Pieter Van Gorp. UML profile for data modeling, *http://www.fots. ua.ac.be/˜pvgorp/research/datamodelingprofile/*, 2007.

[16] IBM Corporation. Rational Data Architect Version 7.0, *http://www-306.ibm.com/software/data/integration/rda/*.

[17] IBM Corporation. Rational Software Architect Version 7.0, *http://www-306.ibm.com/software/awdtools/architect/swarchitect/*.

[18] Robert J. Muller. *Database design for smarties: using UML for data modeling*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[19] MySQL. MySQL 3.23, 4.0, 4.1 Reference Manual, *http://dev.mysql.com/doc/refman/4.1/en/create-table.html*.

[20] No Magic, Inc. MagicDraw UML, *http://www.magicdraw.com*.

[21] Object Management Group (OMG). Request For Proposal Information Management Metamodel (IMM), *http://www.omg.org/docs/ab/05-12-02.pdf*. Technical report, 2005.

[22] Object Management Group (OMG). OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2, *http://www.omg.org/docs/formal/07-11-01.pdf*. Technical report, 2007.

[23] Object Management Group (OMG). UML Profile Catalog, *http://www.omg.org/technology/documents/profile_catalog.htm*. Technical report, 2008.

[24] William J. Premerlani and Michael R. Blaha. An approach for reverse engineering of relational databases. *Commun. ACM*, 37(5):42–49, 134, 1994.

[25] Darius Silingas and Saulius Kaukenas. Applying UML for relational data modeling, *http://www.magicdraw.com/files/articles/Sep04%20Applying%20UML%20for%20 Relational%20Data%20Modeling.htm*, 2004.

[26] Eunjee Song, Shuxin Yin, and Indrakshi Ray. Using uml to model relational database operations. *Comput. Stand. Interfaces*, 29(3):343–354, 2007.

[27] Toby J. Teorey, Dongqing Yang, and James P. Fry. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Comput. Surv.*, 18(2):197–222, 1986.

[28] Hongji Yang and William C. Chu. Acquisition of entity relationship models for maintenance-dealing with data intensive programs in a transformation system. *J. Inf. Sci. Eng.*, 15(2):173–198, 1999.

[29] Shuxin Yin and Indrakshi Ray. Relational database operations modeling with uml. In *AINA '05: Proc. 19th Intl. Conference on Advanced Information Networking and Applications*, pages 927–932, 2005.