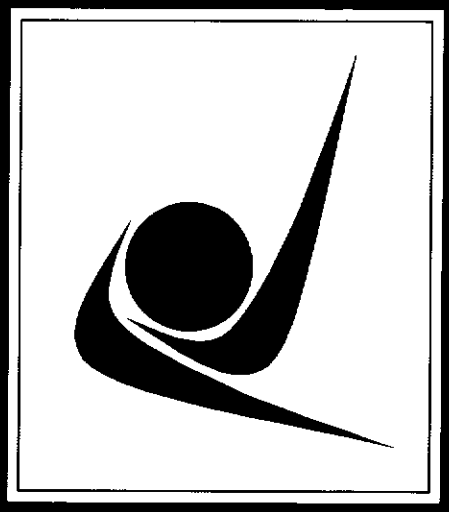

QSIC 2006

Proceedings of the Sixth International Conference on Quality Software

Beijing, China, 27-28 October 2006

Edited by Hong Mei

Organizers:
◆ Peking University, China

Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number P2718
ISBN 0-7695-2718-3
ISSN 1550-6002

Proceedings

//Sixth International Conference on Quality Software//

QSIC//2006

Proceedings

//Sixth International Conference on Quality Software//

QSIC//2006

Los Alamitos, California
Washington • Tokyo

IEEE Computer Society Order Number P2718
ISBN 0-7695-2718-3
ISBN-13 978-0-7695-2718-5
ISSN 1550-6002

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: reprints@computer.org

Editorial production by Patrick Kellenberger

Cover art production by Joe Daigle/Studio Productions

Printed in the United States of America by Applied Digital Imaging

IEEE Computer Society
Conference Publishing Services
http://www.computer.org/proceedings/

//Sixth International Conference on Quality Software//

QSIC//2006

//TABLE OF CONTENTS//

Session 1B//Validation and Verification 1

Session 1C//Software Testing 1

Session 2A// Requirements Engineering

Session 2B// Formal Methods

Session 3A//Quality Management

Session 3B//Software Architecture, Pattern and Framework

Session 3C// Software Testing 2

Session 4A// Quality Attributes Measurement and Analysis 2

Session 4B//Validation and Verification 2

Session 4C//Agile Development and Education

Session 5A//Component-Based Systems

Session 5B//Model Checking

Session 2C// ISEAT 2006

//Message from the General Chair//

It has been a long-waiting event for QSIC, the International Conference on Quality Software, to finally come to Beijing, the capital of China.

It was planned, three years ago, to have QSIC 2003 held in Beijing. But, due to the unfortunate outbreak of SARS, it had to move to Dallas, USA. After that, the conference was hosted in Braunschweig, Germany, in 2004, and in Melbourne, Australia, in 2005. Now it is the time for the young High-Tech conference to meet the ancient cultural city of Beijing.

Beijing is a city which is both old and young. It is old because it has a recorded history dated back to more than 3,000 years ago and the capital for 9 Dynasties. It is young because it keeps growing, both physically and spiritually, almost every day, and because it posses a strong desire to embrace new scientific discoveries and technological innovations which QSIC is devoted to.

Many people have been working hard to make the conference a success. I would like to thank the program committee, especial its chair Hong Mei, for producing the excellent technical program, and the members of the organizing committee, especially Qianxiang Wang and Donggang Cao, for arranging the conference venue, taking care of the financial issues and the conference website. I would also like to thank the steering committee, especially its chair T.H. Tse and member T.Y. Chen, for their invaluable advices and generous supports during the preparation of the conference. Finally, I would like to thank the organizing institution, the Institute of Software of Beijing University, for its strong support to this conference.

Hope you will enjoy the conference and the city of Beijing.

Huimin Lin
Institute of Software, Chinese Academy of Sciences, China

QSIC//2006

//Message from the Program Committee Chair//

Welcome to QSIC 2006, the Sixth International Conference on Quality Software, and to the ancient and young city of Beijing.

Software is becoming ubiquitous. From the tiny sensor to the huge World Wide Web, you can see so many devices are becoming software intensive. While the multifarious runtime environments enhanced the applications of software, and even lead to the service oriented computing in recent years, they are bringing also more dynamic, uncertain, and even error-prone to software. This situation makes software quality a serious problem for a long time. International Conference on Quality Software provides a platform for software researchers to exhibit and exchange their work on software quality, from idea, solution to experience. We hope all attendees can benefit from this platform by joining different sessions and meeting old and new friends!

This year, one hundred eighty one submissions were received. This is just the largest number for QSIC paper submission (In 2003, one hundred eighty submission were received). These papers covering a large variety of topics in quality software: formal methods, testing methods, tools, and education etc. Each submission was assigned to at least three PC members. In the end, fifty papers with high quality were published in the proceeding of QSIC 2006, representing a 28% acceptance rate.

First, we would like to thank the program committee members, who worked so hard to review submitted papers in such a tight time period. We also thank Michael Winikoff, Hong Zhu, and Zhi Jin for organizing the second International Workshop on Integration of Software Engineering and Agent Technology(ISEAT'06). We are indebted to the Steering Committee members, in particular T. Y. Chen, Huimin Lin, and T. H. Tse for their strong support. To be frankly, many important decisions and detailed arrangement are made under their advices. We want thank also Qianxiang Wang, and Donggang Cao, who did many miscellaneous work for the conference.

Because this is the first time that QSIC was held in Beijing. We sincerely hope you enjoy your time these days!

Hong Mei
Peking University, China

QSIC//2006

//Conference Committees//

//STEERING COMMITTEE//

Chair

T.H. Tse, The University of Hong Kong, Hong Kong

Members

T.Y. Chen, Swinburne University of Technology, Australia
Hans-Dieter Ehrich, Technische Universitaet Braunschweig, Germany
Huimin Lin, Institute of Software, Chinese Academy of Sciences, China
Peter C. Poole, The University of Melbourne, Australia
C.V. Ramamoorthy, University of California at Berkeley, USA
Stephen S. Yau, Arizona State University, USA

//GENERAL CHAIR//

Huimin Lin, Institute of Software, Chinese Academy of Sciences, China

//PROGRAM COMMITTEE//

Chair

Hong Mei, Peking University, China

Members

Doo-Hwan Bae, Korean Advanced Institute of Science and Technology, Korea
Elisa Baniassad, The Chinese University of Hong Kong, Hong Kong
Maarten Boasson, University of Amsterdam, The Netherlands
Kai-Yuan Cai, Beijing Univesity of Aeronautics and Astronautics, China
Keith C.C. Chan, The Hong Kong Polytechnic University, Hong Kong
W.K. Chan, The Hong Kong University of Science and Technology, Hong Kong
Carl K. Chang, Iowa State University, USA
Jason Chen, National Central University, Taiwan
Jessica Chen, University of Windsor, Canada
S.C. Cheung, The Hong Kong University of Science and Technology, Hong Kong
William C.-C. Chu, Tunghai University, Taiwan

Takeshi Chusho, Meiji University, Japan
Jin Song Dong, National University of Singapore, Singapore
Yuxi Fu, Shanghai Jiaotong University, China
Kokichi Futatsugi, Japan Advanced Institute of Sci.&Tech., Japan
Arnaud Gotlieb, IRISA-INRIA, France
Wolfgang Grieskamp, Microsoft Research, USA
Xudong He, Florida International University, USA
Zhi Jin, Chinese Academy of Science, China
Ho-Won Jung, Korea University, Korea
Kouichi Kishida, SRA-Key Tech Lab, Tokyo, Japan
Bodgan Korel, Illinois Institute of Technology, USA
Richard Lai, La Trobe University, Australia
Man Fai Lau, Swinburne University of Technology, Australia
Insup Lee, University of Pennsylvania, USA
Shaoying Liu, Hosei University, Japan
Jian Lu, Nanjing University, China
Aditya Mathur, Purdue University, USA
Johannes Mayer, University of Ulm, Germany
Atif M Memon, University of Maryland, USA
Takako Nakatani, University of Tsukuba, Japan
Atsushi Ohnishi, Ritsumeikan University, Japan
Amit Paradkar, IBM T.J.Watson Research Center, USA
Andy Podgurski, Case Western Reserve University, USA
Pak-Lok Poon, The Hong Kong Polytechnic University, Hong Kong
Isidro Ramos, Universidad Politecnica de Valencia, Spain
Per Runeson, Lund University, Sweden
Motoshi Saeki, Tokyo Institute of Technology, Japan
Klaus-Dieter Schewe, Massey University, New Zealand
Tony Shan, Wachovia Bank, USA
Jun Shen, UniSA and UoW, Australia
Paul Strooper, The University of Queensland, Australia
Markus Stumptner, University of South Australia, Australia
Kenji Taguchi, National Institute of Informatics, Japan
J. Barrie Thompson, University of Sunderland, UK
Wei-Tek Tsai, Arizona State University, USA
June Verner, University of New South Wales, Australia
Ji Wang, Changsha Institute of Technology, China
Eric Wong, University of Texas at Dallas, USA
Martin Woodward, University of Liverpool, UK
Min Xie, National University of Singapore, Singapore
Tao Xie, North Carolina State University, USA
Chunxiao Xing, Tsinghua University, China
Baowen Xu, Southeast University, China
Qiwen Xu, University of Macau, Macau
Hongji Yang, de Montfort University, UK
Yuen Tak Yu, City University of Hong Kong, Hong Kong

Jian Zhang, Chinese Academy of Sciences, China
Wenyun Zhao, Fudan University, China
Hong Zhu, Oxford Brookes University, UK

//ORGANIZING COMMITTEE//

Chair

Qianxiang Wang, Peking University, China

Publicity Chair

Donggang Cao, Peking University, China

Members

Feng Chen, University of Illinois at Urbana-Champaign, USA
Rosziati Ibrahim, Universiti Malaysia Sarawak, Malaysia
Xiaodong Liu, Napier University, UK
Lu Zhang, Peking University, China
Katsuhisa Maruyama, Ritsumeikan University, Japan
Robert Merkel, Swinburne University of Technology
Xianping Tao, Nanjing University, China
Jun Wei, Chinese Academy of Sciences, China
Yong Zhang, Tsinghua University, China

//ISEAT 2006 Workshop Committees//

//The Second International Workshop on Integration of Software Engineering and Agent Technology// ISEAT//2006

Workshop Programme Committee Co-chairs

Hong Zhu, Oxford Brookes University, England
Michael Winikoff, RMIT University, Australia
Zhi Jin, Chinese Academy of Sciences, China

Workshop Programme Committee Member

Bernhard Bauer, Universität Augsburg, Germany
Paolo Bresciani, ITC-irst, Italy
Massimo Cossentino, Italian National Research Council, Italy
Frank Dignum, Utrecht University, Netherlands
Scott DeLoach, Kansas State University, USA
Alessandro Garcia, Lancaster University, UK
Aditya K. Ghose, Univ. of Wollongong, Australia
Paolo Giorgini, University of Trento, Italy
Xudong He, Florida International University, USA
Brian Henderson-Sellers, University of Technology Sydney, Australia
Marc-Philippe Huget, University of Savoie, France
Michael Huhns, University of South Carolina, USA
Thomas Juan, University of Melbourne, Australia
Manuel Kolp, Universite catholique de Louvain, Belgium
Ho-fung Leung, Chinese University of HK, China
Jürgen Lind, iteratec GmbH, Germany
Lin Liu, Tsinghua University, China
Graham Low, University of New South Wales, Australia
James Odell, Intelligent Automation Inc, USA
Andrea Omicini, Università di Bologna a Cesena, Italy
Onn Shehory, IBM, Israel

Leon Sterling, University of Melbourne, Australia
Arnon Sturm, Ben-Gurion University of the Negev, Israel
Gerhard Weiss, SCCH GmbH, Austria
Dianxiang Xu, North Dakota State University, USA
Manwu Xu, Nanjing University, China
Hongji Yang, DeMonte Fort University, UK

Object-Relational Database Metrics Formalization

Aline Lúcia Baroni[1], Coral Calero[2], Fernando Brito e Abreu[1], Mario Piattini[2]
1Universidade Nova de Lisboa (New University of Lisbon)
QUASAR Research Group / CITI / DI / FCT
Quinta da Torre, 2829-516, Monte da Caparica – Portugal
2Universidad de Castilla-La Mancha (University of Castilha-La Mancha)
ALARCOS Research Group / Escuela Superior de Informática de Ciudad Real
Paseo de la Universidad, 4, 13071 Ciudad Real - Spain
{alinebaroni, fba}@di.fct.unl.pt; {coral.calero, mario.piattini}@uclm.es

Abstract

The abstract is to be in fully-justified italicized text, at the top of the left-hand column as it is here, below the author information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type, and up to 150 words in length. Leave two blank lines after the abstract, then begin the main text.

1. Introduction

The history of databases has been characterized by its extraordinary productivity and its impressive economic impact. They have become a strategic product, being the basis of information systems and supporting organizational decisions.

Relational databases are the most important ones in the database world. This success can be explained, among others, by the next reasons:

(i). they are easy to understand;
(ii). there is a well-known standard (SQL) supporting them;
(iii). they are built on proven theoretical foundations that have stood the test of time;
(iv). they have been proven in industry over many years and are installed into many businesses worldwide;
(v). they already have millions of lines of code written for them and
(vi). object-relational databases (ORDBs) introduced the possibility of working with new and complex data and applications, without a revolutionary change in the market.

ORDBs include all the elements of the relational model (relations connected by referential integrity relationships) but with the particularity that the columns of a relation can be defined over a *user defined type*. Some studies predict that ORDBs will substitute the relational ones [1, 2]. In fact, since the version of 1999, the SQL standard (SQL, 1999) used by most of the Relational Database Management Systems (RDBMs), implements the object-relational model, and even in the last version of the standard, the SQL:2003, more object aspects have been included (SQL, 2003).

Taking into account the future predicted for the object-relational databases in the database market and to ensure that it becomes a reality, it is essential to have "good" designs. One widely accepted mechanism for assuring the quality of a software product in general and of object-relational database designs in particular, is the use of metrics specifically designed for this kind of databases. Software metrics can be used to build prediction systems for database projects [3], to understand and improve software development and maintenance projects [4], to highlight the system problematic areas [5] and to guide developers and researchers in their work [6]. It is also possible to use the metrics for reducing the database maintenance effort, applying their formal definitions to database schemas in order to guide and assess transformations that can improve quality, by reducing schema's complexity.

There are hundreds of metrics defined for software products, processes and resources [6, 7]. However, most of them do not go beyond the definition stage, being not applied on industry. This can be explained by the ambiguity on the definitions and assumptions on the metrics that entail difficult usage, dangerous interpretations and contradictions on the validation studies. A good way of avoiding this shortcoming is through formalization. Formality enables to have clear metrics definitions, which in turn assures that their computation can be repeated in a reliable fashion.

0-7695-2718-3/06 $20.00 © 2006 IEEE

Furthermore, the formalization itself may facilitate the automation of metrics collection.

In this paper the formalization of a set of metrics for assessing the complexity of ORBDs is presented. An ontology for the SQL:2003 standard [8] was produced, as a framework for representing the SQL schema definitions. It was represented using UML [9]. The metrics were defined with OCL [10], upon the SQL:2003 ontology. The remainder of this paper is organized as follows: section 2 briefly presents the ontology we have defined for the SQL:2003. Section 3 illustrates an example of a database definition, represented using the ontology. Section 4 shows the metrics, their formalization and the results for the example on section 3. Finally, section 5 outlines the conclusions and future work.

2. SQL:2003 ontology

Among the different languages present in the earliest DBMS, SQL imposed itself as a *de jure* and *de facto* standard. In fact, the SQL is used as basis for most of the object-relational DBMS. Recently, the SQL:2003 standard was published [11]. It makes revisions to all the parts of SQL:1999 [12] and includes some new elements [13].

Having a standard is fundamental. Conversely, sometimes standards are hard to understand and it is difficult to extract all the information contained on it. It usually happens that standards are not free from inconsistencies, due to the large amount of information they cover. In that case, some advantages derived from the usage of a standard disappear. For preventing these problems, the standard can be complemented with an ontology. In such a manner, the ontology helps to find the information out and to detect inconsistencies.

We developed an ontology for the SQL:2003, using several parts of this textual standard. We worked mainly with the information of the parts 1 (Framework), 2 (Foundation) and 11 (Information and Definition Schema). The ontology was divided in two building blocks. One contains all the aspects related to data types (Figure 1) and the other all the information about the SQL schema objects (Figure 2). The ontology has been represented in Rational Rose using UML and the complete description can be found on [14].

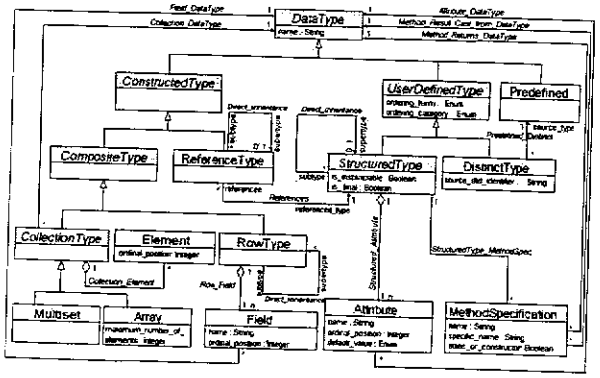

Figure 1. Data Types sub-ontology

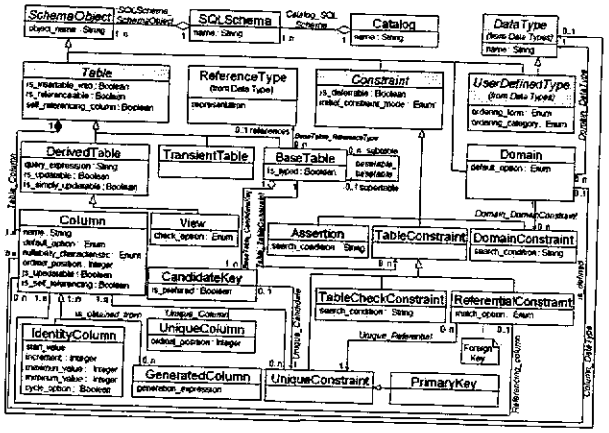

Figure 2. Schema Object sub-ontology

In figure 3 we exemplify how to instantiate the ontology to represent an ORDB schema. The example corresponds to the SQL:2003 code of table 1 which is based on the example from [15].The schema is composed of four tables (*customers*, *music_distributors* which is typed, *movie_stars* and *movies*). The *customers* table has six simple columns (one of them is an identity column and another is a generated column) and one complex column. The *movie_stars* table has six simple columns and the *movies* table has three simple columns and one complex column. The schema has defined primary keys, foreign keys, constraints, distinct types, collection types, row types, an assertion and a view. Figure 3 elucidates the representation of the SQL:2003 code (Table 1), using the ontology notation. The figure has been automatically generated from the SQL code, mapping the code with the SQL:2003 ontology representation.

0-7695-2718-3/06 $20.00 © 2006 IEEE

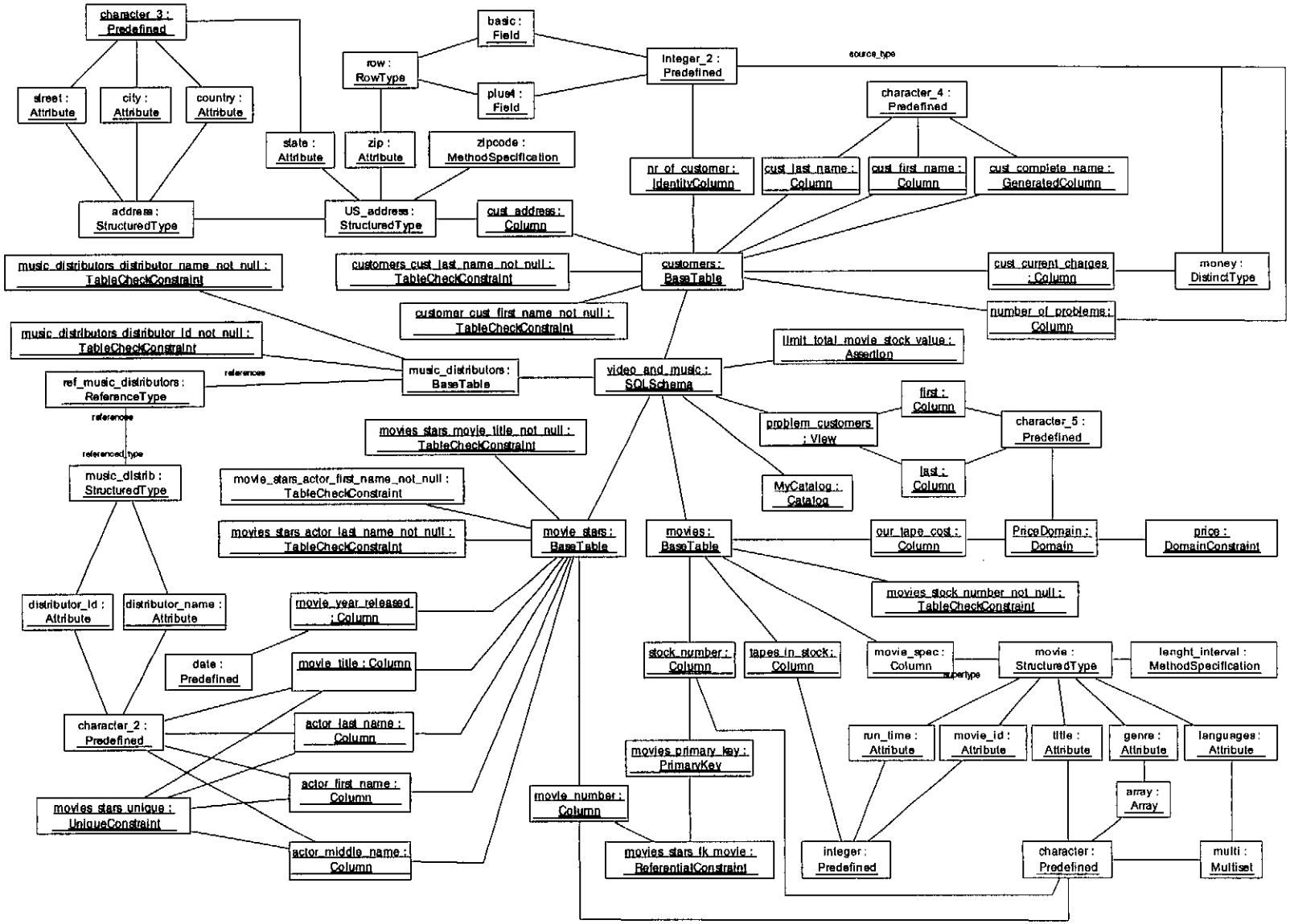

Figure 3. Instantiation example

```
CREATE SCHEMA video_and_music
AUTHORIZATION m_s_enterprises
DEFAULT CHARACTER SET "Latin_1"

CREATE DOMAIN price DECIMAL (7,2)
CHECK (VALUE IS NOT 0);

CREATE DISTINCT TYPE money AS DECIMAL (9,2);

CREATE TYPE movie AS(
  movie_id INTEGER,
  title CHARACTER VARYING (100),
  languages MULTISET
          ['English', 'French', 'Spanish',
           'Portuguese', 'Italian'],
  genre CHARACTER VARYING (20) ARRAY [10],
  run_time INTEGER)
INSTANTIABLE NOT FINAL
METHOD length_interval ()
RETURNS INTERVAL HOUR (2) TO MINUTE

CREATE INSTANCE METHOD length_interval ()
RETURNS INTERVAL HOUR (2) TO MINUTE FOR MOVIE
RETURN CAST (CAST (SELF.run_time AS INTERVAL (4))
AS INTERVAL HOUR (2) TO MINUTE);

CREATE TYPE music_distrib AS (
  distributor_id CHARACTER (15),
  distributor_name CHARACTER (25));

CREATE TYPE address AS(
  street CHARACTER VARYING (35),
  city CHARACTER VARYING (40),
  country character (3));

CREATE TYPE US_address UNDER address AS(
  state CHARACTER (2),
  zip   ROW ( Basic INTEGER, Plus4 SMALLINT))

METHOD zipcode ()
RETURNS CHARACTER VARYING (10);

CREATE INSTANCE METHOD zipcode ()
RETURNS CHARACTER VARYING (10) FOR US_address
BEGIN
IF SELF.zip.plus4 IS NULL
THEN RETURN CAST (SELF.zip.basic AS
    CHARACTER VARYING (5));
         ELSE RETURN CAST (SELF.zip.basic AS
              CHARACTER VARYING (5))
    || '-'  || CAST (SELF.zip.basic AS
    CHARACTER VARYING (4))
ENDIF;
END;

CREATE TABLE movies (
  stock_number CHARACTER(10)
    CONSTRAINT movies_stock_number_not_null NOT NULL,
  movie_spec movie,
  our_tape_cost price,
  tapes_in_stock INTEGER
    CONSTRAINT movies_primary_key
    PRIMARY KEY (stock_number));
```

```
CREATE TABLE movie_stars (
  movie_title CHARACTER (30)
    CONSTRAINT
     movies_stars_movie_title_not_null NOT NULL,
  movie_year_released DATE,
  movie_number CHARACTER (10),
  actor_last_name CHARACTER (35)
    CONSTRAINT
     movies_stars_actor_last_name_not _null NOT NULL,
  actor_first_name CHARACTER (25)
    CONSTRAINT
     movies_stars_actor_first_name_not _null
    NOT NULL,
  actor_middle_name CHARACTER (25),
    CONSTRAINT movies_stars_unique
    UNIQUE (movie_title, actor_last_name,
            actor_first_name,
            actor_middle_name)
     NOT DEFERRABLE,
     CONSTRAINT movies_stars_fk_movie
     FOREIGN KEY (movie_number)
          REFERENCES movies (stock_number)
          ON DELETE CASCADE
            ON UPDATE CASCADE);

CREATE TABLE music_distributors
       OF music_distributors (
REF IS dist_ref SYSTEM GENERATED,
  distributor_id WITH OPTIONS CONSTRAINT
   music_distributors_distributor_id_not_null
   NOT NULL,
  distributor_name WITH OPTIONS CONSTRAINT
 music_distributors_distributor_name_not_null
   NOT NULL, );

CREATE TABLE customers(
 nr_of_customer INTEGER GENERATED ALWAYS
                AS IDENTITY (START WITH 1
                INCREMENTED BY 1MINVALUE 1),
 cust_last_name CHARACTER (35)
  CONSTRAINT customers_cust_last_name_not_null
  NOT NULL,
 cust_first_name CHARACTER (35)
  CONSTRAINT customers_cust_first_name_not_null
  NOT NULL,
 cust_complete_name GENERATED ALWAYS AS
        (cust_first_name || cust_last_name),
 cust_address US_address,
 cust_current_charges money,
 number_of_problems SMALLINT);

CREATE VIEW problem_customers (last, first) AS
  SELECT cust_last_name, cust_first_name
  FROM  customers
  WHERE number_of_problems <
  0.8 * (SELECT MAX(number_of_problems)
        FROM customers);

CREATE ASSERTION
  limit_total_movie_stock_value
  CHECK (( SELECT COUNT(*)
           FROM customers
           WHERE number_of_problems > 5
           AND cust_current_charges > 150,00
           AND cust_current_charges <
           1000,00) <10);
```

Table 1. SQL:2003 example

3. Measuring object-relational databases

Metrics can be defined to capture a specific attribute value of a product (in our case object-relational databases). One of the most important characteristics to be captured is complexity. With a set of metrics for measuring the complexity, we are able to estimate understandability and maintainability [16-18], two important dimensions in software product quality [8].

3.1. Metrics for object-relational databases

In [19] a set of metrics for object-relational database complexity was defined. The metrics were validated as good indicators of the complexity of an object-relational database. The validation was done both formally and empirically through controlled experiments. The formalization of those metrics using the approach presented in [20, 21], is shown on the next paragraphs. For each metric, an informal definition is presented together with its formal one (an OCL expression). Some auxiliary functions used in the formalization process are presented after the metrics formalization.

3.2. Size Metrics

The size of a table (TS) is defined as the sum of the size of the simple columns (TSSC) and the size of the

complex columns (TSCC). The TSCC is calculated as the sum of the size of each complex column (CCS).

```
BaseTable:: TS(): Real =
if self.is_typed then
self.references.referenced_type.
      hierarchySize()
else self.TSCC() + self.TSSC()
endif

BaseTable:: TSSC(): Integer =
self.allSimpleColumns() -> size()

BaseTable:: TSCC(): Real =
  self.allComplexColumns()->
   collect(elem: Column |elem.CCS()) -> sum
```

The size of a complex column (CCS) is defined as the size of the class hierarchy above which the column is defined (SHC) divided by the number of complex columns that are defined over this hierarchy (NHC). This expression is due to the fact that the size of the hierarchy must be considered only once independently of the number of columns defined above it.

```
Column:: CCS(): Real = self.SHC() / self.NCU()

Column:: SHC(): Real =
self.dataType.oclAsType(StructuredType).SC() +
self.dataType.oclAsType(StructuredType)
    .ascendants()->
       collect (elem: DataType |
       elem.oclAsType(StructuredType).SC()) ->
              sum

Column:: NCU(): Integer =
self.dataType.oclAsType(StructuredType).column
sNumberUsingThis()
```

The size of a class (SC) is calculated as the sum of its attributes size (SAC) and its methods size (SMC). It is necessary to take into account that a class can have simple attributes (SAS), which have a size equal to one, and complex attributes (CAS), which are attributes related to other classes by an aggregation relationship. In that case the size of a complex attribute is calculated as the size of the aggregation hierarchy. Again in that case, as a class can belong to more than one hierarchy, it is necessary to divide its size into the number of hierarchies that use the class (NHC).

```
StructuredType:: SC() : Real =
(self.SAC() + self.SMC()) / self.NHC()

StructuredType:: SAC(): Real =
self.SAS() + self.CAS()

StructuredType:: SAS(): Integer =
self.allSimpleAttributes() -> size()

StructuredType:: CAS(): Real =
self.allComplexAttributes()
-> collect(elem: Attribute |
elem.dataType.oclAsType(StructuredType).SC())
    -> sum

StructuredType:: SMC(): Integer = self.NMC()

StructuredType:: NMC(): Integer =
self.allMethods() -> size()

StructuredType:: NHC(): Integer =
if self.hasChildren() then
      self.childrenNumber()
else 1 endif
```

PCC (Percentage of Complex Columns): Number of the complex columns of a table (NCC) divided by the total number of columns of the same table.

```
BaseTable:: PCC(): Percentage =
self.NCC()/(self.allColumns()-> size())

BaseTable:: NCC(): Integer =
self.allComplexColumns() -> size()
```

3.3. Coupling Metrics

NIC (Number of Involved Classes): Number of classes needed for defining all the columns of a table.

```
BaseTable:: NIC(): Integer =
self.involvedClasses() -> size
```

NSC (Number of Shared Classes): Number of classes used by a table, for defining its complex columns, which are also used by other tables of the schema.

```
BaseTable:: NSC(): Integer =
self.involvedClasses()
    -> select(elem: StructuredType |
            elem.isShared()) -> size
```

3.4. Referential Integrity Metrics

NFK (Number of Foreign Keys): Number of foreign keys defined in a table.

```
BaseTable:: NFK(): Integer =
self.foreignKeyNumber()
```

RD (Referential Degree): Number of foreign keys in a table divided by the number of attributes of the same table.

```
BaseTable:: RD(): Real =
self.NFK() /(self.allColumns() -> size())
```

DRT (Depth of Referential Tree): The longest path between a table and the remaining tables in the schema database, considering the schema as a graph where nodes are tables and arcs are referential integrity relations between tables (Foreign key to Primary key links).

```
BaseTable:: DRT(): Integer =
self.longestPath() -> size
```

3.5. Generalizing the Collection Process to the Schema

All the metrics shown before are applied over tables and can also be applied to the schema level, iterating over the *BaseTables* in the *SQLSchema*. For example, a size metric for the schema (SS) can also be defined, as the sum of the sizes of each table in the schema.

```
SQLSchema:: SS(): Real =
self.allBaseTables()-> collect
(elem: BaseTable |elem.TS()) -> sum
```

3.6. Auxiliary Functions

Many of auxiliary functions were employed on the metrics definitions, as navigations upon ontology entities. Some of them are explained below (we do not

introduce all due to space restrictions) in the order they appear in the above sub-sections.

***3.6.1. Function name*: hierarchySize()**

Informal definition: Size of the hierarchy above which the StructuredType is defined.

Formal definition:

```
StructuredType:: hierarchySize(): Real =
self.SC() + self.ascendants()
  -> collect(elem: StructuredType | elem.SC())
     -> sum
```

***3.6.2. Function name*: allSimpleColumns()**

Informal definition: Set of simple Columns belonging to the current BaseTable. A simple Column is a Column whose type is neither StructuredType nor ReferencedType.

Formal definition:

```
BaseTable:: allSimpleColumns() : Set(Column) =
self.allColumns() ->
reject(domain.dataType.
    oclIsTypeOf(StructuredType)
or domain.dataType.oclIsTypeOf(ReferenceType)
or dataType.oclIsTypeOf(ReferenceType)
     or dataType.oclIsTypeOf(StructuredType))
```

***3.6.3. Function name*: allComplexColumns()**

Informal definition: S Set of complex Columns belonging to the current BaseTable. A complex Column is a Column whose type is either StructuredType or ReferencedType.

Formal definition:

```
BaseTable:: allComplexColumns() : Set(Column)
=self.allColumns() - self.allSimpleColumns()
```

***3.6.4. Function name*: ascendants()**

Informal definition: Set of all classes from which the current StructuredType derives (both directly and indirectly).

Formal definition:

```
StructuredType:: ascendants() :
Set(StructuredType) =
self.parents()
  -> collect(elem: StructuredType
     | elem.ascendants()
      -> union(self.parents()))
         -> asSet() -> flatten
```

***3.6.5. Function name*: allSimpleAttributes()**

Informal definition: Set of simple Attributes declared in the current class. Simple Attributes are the ones whose type is neither a StructuredType nor a ReferenceType.

Formal definition:

```
StructuredType::
allSimpleAttributes(): Set(Attribute) =
self.allAttributes()→
reject(dataType.oclIsTypeOf(StructuredType)
     or dataType.oclIsTypeOf(ReferenceType))
```

***3.6.6. Function name*: allComplexAttributes()**

Informal definition: Set of complex Attributes declared in the current class. Complex Attributes are the ones whose type is StructuredType.

Formal definition:

```
StructuredType::
allComplexAttributes() : Set(Attribute) =
self.allAttributes()->
select(dataType.oclIsTypeOf(StructuredType))
```

***3.6.7. Function name*: allMethods()**

Informal definition: Set of methods in the current class without considering inheritance.

Formal definition:

```
StructuredType:: allMethods():
Set(MethodSpecification) =
self.methodSpecification
```

***3.6.8. Function name*: hasChildren()**

Informal definition: Indicates whether the StructuredType has children or not. A true value indicates it has children and a false value indicates the contrary.

Formal definition:

```
StructuredType:: hasChildren() : Boolean =
self.childrenNumber() > 0
```

***3.6.9. Function name*: childrenNumber()**

Informal definition: Number of directly derived classes.

Formal definition:

```
StructuredType:: childrenNumber(): Integer =
self.children() -> size()
```

***3.6.10. Function name*: involvedClasses()**

Informal definition: Set of classes involved with the current one.

Formal definition:

```
BaseTable::involvedClasses():
Set(StructuredType)
= self.complexColumnTypes()
  -> union (self.allComplexColumns()
     -> collect(c: Column |
        c.columnType().
       oclAsType(StructuredType).
        allDependencies())
          -> flatten) -> asSet
```

3.6.11. Function name: isShared()

Informal definition: Indicates whether the current StructuredType is shared among different BaseTables. Returns true when shared, false otherwise.

Formal definition:

```
StructuredType:: isShared(): Boolean =
self.children()-> iterate(elem: DataType;
acc: Boolean = self.tablesNumberUsingThis() >
   1 | Acc or
    elem.oclAsType(StructuredType).isShared())
```

***3.6.12. Function name*: foreignKeyNumber()**

Informal definition: Number of foreign keys a BaseTable has.

Formal definition:

```
BaseTable:: foreignKeyNumber():Integer =
self.allColumns()->
select(elem: Column | elem.hasForeignKey())
 -> collect(elem: Column |
    elem.foreignKeyNumber()) -> sum
```

***3.6.13. Function name*: longestPath()**

Informal definition: Set of BaseTables which forms the longest path from the current BaseTable to other one, related to the current through one or more Constraints.

Formal definition:

```
BaseTable:: longestPath(): Set(BaseTable)=
self.comparePaths(oclEmpty(Set(BaseTable)),
oclEmpty(Set(BaseTable))) -> excluding(self)
```

3.7. Metrics Values for the Schema Example

Tables 2, 3 and 4 exemplify the metrics results when applying the previous formalization expressions to the code of Table 1, and represented using the ontology notation in Figure 3.

	Cust_Address	*Movie_Spec*
SHC	3	6
NCU	1	1
CCS	3	6

Table 2. SHC, NCU and CCS Metric values

All the OCL expressions showed previously (for implementing the metrics) accompany the ontology UML diagram in such a manner that they are calculated on a concrete example automatically.

The SS value for video_and_music is the sum of TS for the BaseTables in Table 4, i.e.: SS = 9 + 9 + 5 + 2 = 25.

	US_Address	*Address*	*Movie*
SAS	2	3	5
CAS	0	0	0
SAC	2	3	5
SMC	1	0	1
NHC	1	1	1
SC	3	3	6

Table 3. Metric Values

	Customers	*Movies*	*Movie Stars*	*Music Distributors*
TSSC	6	3	5	-
TSCC	3	6	0	-
TS	9	9	5	2
NIC	1	1	0	1
NSC	0	0	0	0
PCC	0.14	0.25	0	0
NCC	1	1	0	0
NFK	0	0	1	0
RD	0	0	0.2	0
DRT	0	0	1	0

Table 4. Rest of metrics

These metrics values could be used in several ways. For example, the metrics can be applied to a particular database schema design and, using a set of threshold values, the designer could decide if the design has a complexity that fits into an acceptable level or not (in the last case, the design must be changed using the metrics for detecting where it is better to made the changes). Another possible utility of the metrics is the comparison of two database schema design semantically equivalent. The metrics are then used for deciding which one is less complex, being this one the selected for the implementation. Other possible use of the metrics arises a a result of the database lifetime because the initial design suffers continuous evolution. Over subsequent versions there is a risk that the design becomes more complex and less maintainable, reducing the overall system quality. In this case, database schema refactorings must to be used. The metrics can be used for evaluating a database schema to identify tables with higher complexity, which hampers their understandability and maintainability. The appropriate refactoring(s) can be then applied to those tables. After applying the refactoring OR database metrics are collected again upon the refactored tables and the results are compared with those collected first. If these metrics do not grant evidence that the complexity of the refactored-tables has indeed dropped down, then a rollback is advised. Otherwise, in case of complexity reduction, the changes should be propagated to all interested parties (technical team) and should be kept consistent with other deliverables.

4. CONCLUSIONS

Our current work direction addresses the solution of two main problems: the lack of metrics for evaluating the quality of databases and the lack of formalization of the existing metrics definitions. The first problem was treated with the proposal of some metrics for object-relational databases [19], in our previous work. However, metrics for different aspects, not covered by

our work, are still necessary. This paper presented an approach to solve the second problem, using UML and OCL [9, 10, 22]. The original approach was proposed in [20], and it was successfully applied here.

Besides formalizing some metrics definitions, we created an ontology for the new SQL:2003 standard, which tries not only to reduce the inconsistencies in the textual version of the standard, but also to make the standard easier to grasp [14]. The formalized metrics definitions and a database representation mapped to ontology meta-objects served as input to an OCL evaluator tool. With these two inputs, and also with the ontology as background, one can extract real metric values from database representations. One simple example was illustrated here. As future work, there are many possible directions to explore, varying from the proposition of new metrics and their validation, including their formal definitions, until the use of these metrics to perform refactorings on database schemata. We started to investigate the latter [23].

5. ACKNOWLEDGMENTS

This work was partly funded by the Portuguese Foundation of Science and Technology (FCT – MCES: http://www.fct.mces.pt/) and by projects CALIPO (TIC2003-07804-C05-03) and CALIPSO (TIN24055-E) from the MEC (Spain).

6. REFERENCES

[1] M. Stonebraker and P. Brown, Object-Relational DBMSs Tracking the Next Great Wave. California: Morgan Kaufmann, 1999.

[2] N. Leavitt, "Whatever Happened to Object-Oriented Databases?" IEEE Computer Society, Industry Trends, pp. 16-19, 2000.

[3] S. MacDonell, M. Shepperd, and P. Sallis, "Metrics for Database Systems: An Empirical Study," Proceedings of the 4th International Software Metrics Symposium (Metrics' 97), Albuquerque, 1997.

[4] L. C. Briand, S. Morasca, and V. R. Basili, "Property-Based Software Engineering Measurement," IEEE Transactions on Software Engineering, vol. 22, pp. 68-86, 1996.

[5] D. Champeaux, Object-Oriented Development Process and Metrics. Upper Saddle River: Prentice-Hall, 1997.

[6] N. E. Fenton and S. L. Pfleeger, "Software Metrics: A Rigorous & Practical Approach," 2nd ed. London, United Kingdom: International Thomson Computer Press, 1997.

[7] A. Melton, Software Measurement. London, United Kingdom: Thomson Computer Press, 1996.

[8] ISO9126, "ISO/IEC 9126: Information Technology - Software Product Evaluation - Software Quality Characteristics and Metrics." Geneva, Switzerland: International Organization for Standardization, 2001.

[9] OMG, "Unified Modeling Language: Superstructure - Version 2.0 - Final Adopted Specification," Object Management Group Inc. ptc/03-08-02, 2003 2003.

[10] OMG, "Unified Modeling Language: OCL (version 2.0)," Object Management Group Inc. ptc/03-08-08, August 2003.

[11] ISO9075, "ISO/IEC 9075 Standard: Information Technology - Database Languages - SQL," International Organization for Standardization, 2003.

[12] ISO9075, "ISO/IEC 9075: Information Technology - Database languages - SQL," International Organization for Standardization, 1999.

[13] A. Eisenberg, J. Melton, K. Kulkarni, J. Michels, and F. Zemke, "SLQ:2003 Has Been Published," Sigmod Record, vol. 33, pp. 119 - 126, 2004.

[14] C. Calero, F. Ruiz, A. L. Baroni, F. B. Abreu, and M. Piattini, "An Ontological Approach to Describe the SQL:2003 Object-Relational Features", Accepted in Computer Standards and Interfaces, 2006. Volume 29. Issue 1. In press

[15] J. Melton, Advanced SQL:2003 - Understanding Object-Relational and Other Features. USA: Morgan Kaufmann, 2003.

[16] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," Journal of Systems and Software, vol. 23, pp. 111-122, 1993.

[17] L. Briand, K. El Emam, and S. Morasca, "Theoretical and Empirical Validation of Software Product Measures," International Software Engineering Research Network, Technical Report ISERN-95-03, 1995.

[18] L. Briand, E. Arisholm, S. Counsell, F. Houdek, and P. Thévenod-Fosse, "Empirical Studies of Object-Oriented Artifacts, Methods and Processes: State of the Art and Future Directions," Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, Technical Report IESE 037.99/E, 1999.

[19] M. Piattini, C. Calero, H. Sahraoui, and H. Lounis, "Object-Relational Database Metrics," L'Objet, vol. 3, 2001.

[20] A. L. Baroni, "Formal Definition of Object-Oriented Design Metrics," in Computer Science. Brussels, Belgium: Vrije Universiteit Brussel - Belgium, 2002.

[21] A. L. Baroni, S. Braz, and F. B. Abreu, "Using OCL to Formalize Object-Oriented Design Metrics Definitions," presented at QAOOSE'2002, Malaga, Spain, 2002.

[22] OMG, "UML 2.0 Infrastructure Final Adopted Specification," Object Management Group, Inc. ptc/03-09-15, September 2003.

[23] A. L. Baroni, F. B. Abreu, and C. Calero, "Finding Where to Apply Object-Relational Database Schema Refactorings: an Ontology-Guided Approach", X Jornadas sobre Ingeniería del Software y Bases de Datos (JISBD 2005), Granada, Spain, 2005. ISBN: 84-9732-434-X