Proceedings

Sixteenth Working Conference on

Reverse Engineering

13-16 October 2009
Lille, France

Sponsored by
Reengineering Forum

Technical co-sponsor
IEEE Computer Society Technical Council on Software Engineering (TCSE)

IEEE computer society

IEEE

Edited by
Andy Zaidman
Giuliano Antoniol
Stéphane Ducasse

Working Conference on Reverse Engineering 2009

Computer Society Press

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

IEEE Computer Society Order Number P3867
BMS Part Number: CFP09090-PRT
ISSN Number 1095-1350
ISBN 978-0-7695-3867-9

ISBN 978-0-7695-3867-9
90000
9 780769 538679

Proceedings

16th Working Conference

on Reverse Engineering (WCRE 2009)

Proceedings

16^{th} Working Conference on Reverse Engineering (WCRE 2009)

13^{th} – 16^{th} October 2009 – Lille, France

Edited by

Andy Zaidman, Giuliano Antoniol and Stéphane Ducasee

Sponsored by

Reengineering Forum

Technical Co-sponsor

IEEE-CS Technical Council on Software Engineering

In cooperation with

INRIA Lille-Nord Europe

SOCCER LAB
SOftware Cost-effective Change and Evolution Research Lab

TU Delft, Delft University of Technology

CPS Conference Publishing Services

Los Alamitos, California
Washington • Tokyo

IEEE computer society

IEEE Computer Society Order Number P3867
BMS Part Number CFP09090-PRT
ISBN 978-0-7695-3867-9
ISSN Number 1095-1350

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 Bob Werner
Cover art production by Joe Daigle/Studio Productions
Printed in the United States of America by The Printing House

IEEE Computer Society
Conference Publishing Services **(CPS)**
http://www.computer.org/cps

2009 16th Working Conference on Reverse Engineering

WCRE 2009

Table of Contents

Session II – Dynamic Analysis

Session III – Empirical Software Engineering

Session IV – Remodularization and Reengineering

Session V - Change and Defect Proneness

Session VI – Static Analysis and Security

Session VII – Traceability

Session VIII - Program Comprehension

Session IX – Static Analysis

PhD Forum

Tool Demonstrations

Workshops

2009 16th Working Conference on Reverse Engineering

On the use of ADM to Contextualize Data on Legacy Source Code for Software Modernization

Ricardo Pérez-Castillo, Ignacio García-Rodríguez de Guzmán and Mario Piattini
Alarcos Research Group, University of Castilla-La Mancha
Paseo de la Universidad, 4 13071,
Ciudad Real, Spain
{ricardo.pdelcastillo, ignacio.grodriguez, mario.piattini}@uclm.es

Orlando Ávila-García
* Open Canarias, S.L.
C/ Elías Ramos González, nº 4 - Oficina 304 38001,
Santa Cruz de Tenerife, Spain
orlando@opencanarias.com

***Abstract*—Legacy systems are usually made of two kind of artifacts: source code and databases. Typically, the maintenance of those systems is carried out through re-engineering processes. Although both artifacts can be independently maintained, for a more effective re-engineering of the whole system both should be analyzed and evolved jointly. This is mainly due to the fact that the knowledge expected to be extracted by analyzing both kind of artifacts at the same time is greater and richer than the one recovered by just looking at the system partly, and thus ROI and lifespan of the system are expected to improve. This paper proposes the Data Contextualization for recovering code-to-data linkages in legacy systems. This technique is framed in the ADM (Architecture Driven Modernization) approach to modernization of legacy systems, considering all involved artifacts as models. This paper also presents a tool to support that technique throughout a real-life case study.**

Keywords—Data Contextualization, Modernization, Model Transformations, ADM and KDM.

I. Introduction

At the present time, the majority of organizations have large legacy systems supported by relational databases. These systems are not immune to software ageing. The erosion not only affects to the source code, but databases also age gradually. For instance, in order to adapt the system to new requirements, new tables and/or columns are added to the database; other tables are modified and even discarded without erasing them from the database. These changes over time generate problems related to inconsistency, redundancy and integrity among others.

Therefore, organizations must address maintenance processes taking into account legacy source code and databases together. In those maintenances, the entire replacement of the legacy system would have a great technological, strategical and economical impact for the organization. [10]. In addition, according to [2], the 78% of maintenance changes are corrective or behaviour-preserving. Indeed, maintenances based on evolutionary reengineering processes are typically carried out.

The starting point in that reengineering process is the conceptual representation of the legacy system through reverse engineering [1]. At this stage, the legacy source code as well as the legacy database must be represented in order to consider these two artefacts jointly. Nevertheless, a challenge appears in this scenario: finding out what fragments of the database are used by each piece of legacy source code. This knowledge is essential in later stages of the reengineering process, such as restructuring and forward engineering, where the new and improved systems are built [10]. Since the improved system will probably use the same data, it turns out to be very important to keep track of the use the data in the context of the source code of the legacy system. That is to say, it is important to contextualize the data in the representation of the legacy software when we are reversing it.

This paper proposes the *Data Contextualization* technique and a tool that supports it. This is a novel reverse engineering technique developed in the context of the *MARBLE* framework to modernize legacy systems [9]. This technique recovers *code-to-data* links in legacy systems based on relational databases and allows representing and managing these linkages throughout entire reengineering processes. In order to obtain these linkages two main knowledge sources are considered: (i) database schemas and (ii) the sentences embedded in source code. Moreover, the proposal follows ADM (Architecture-Driven Modernization) approach for developing this technique [6]. This approach advocates modelling all artefacts involved in the reengineering process as models and it transforms the models between different abstraction levels according to MDA (Model-Driven Architecture) principles [5].

The remainder of this paper is organized as follows. Section 2 presents the background of this work. Section 3 shows the proposed *Data Contextualization* technique. Section 4 presents the developed tool. Section 5 presents a case study of a real-life modernization project. Finally, section 6 addresses the conclusions and the future work.

II. Background

A. Related work

The inspection of source code and recovery of specific knowledge is a common challenge in reengineering and maintenance processes. Nevertheless, the linkage between code and used data has not been widely studied. *Zou* developed a framework based on a set of heuristic rules for

1095-1350/09 $26.00 © 2009 IEEE
DOI 10.1109/WCRE.2009.20

extracting business processes following a MDA approach [11]. Those works took into account legacy source code and program data in order to link pieces of code together and to obtain workflows. However, they do not link source code and external data such as databases. Some works such as [3] propose frameworks to align and develop business rules by means of collecting information about *how* and *where* these business rules are implemented within the source code. But data and code are not mapped together. *Marinescu* proposes in [4] an approach for determining the correlation between foreign keys extracted from the database schema and the way the data are used in the source code. Finally, there have been research in database reengineering follows the MDA approach [9], but source code is not considered jointly. In spite of these works, in any case the *code-to-data* linkage is carried out following the ADM approach.

B. Architecture-Driven Modernization

Reengineering and MDA have converged on ADM, another OMG initiative. ADM is the concept of modernizing existing systems with a focus on all aspects of the current systems architecture and the ability to transform current architectures to target architectures [6].

The increasing cost of maintaining legacy systems together with the need to preserve business knowledge has turn modernization of legacy systems into an important research field. ADM provides several benefits such as ROI improvement on existing information systems, reducing development and maintenance cost, extending life cycle of the legacy systems, and easy integration with other systems.

ADM Task Force in OMG has led to several standards. The cornerstone within this set of standards is KDM (*Knowledge Discovery Meta*). KDM allows standardized representation of knowledge extracted from legacy systems by means of reverse engineering [8]. KDM provides a common repository structure that makes possible the exchange of information about existing software assets in legacy systems. This information is currently represented and stored independently by heterogeneous tools focused on different software assets. KDM can be compared with the UML (Unified Modeling Language) standard: UML is used to generate new code in a top-down manner. In contrast, a process involving KDM (as *Data Contextualization*) starts from the existing code and builds a higher level model in a bottom-up manner.

ADM and KDM advocate representing any artefact as models according to the MDA approach. Transformations among these models are modelled by means of QVT (Queries / Views / Transformations) [7].

III. DATA CONTEXTUALIZATION

The proposed *Data Contextualization* is a technique that can be used in modernization processes when the reverse engineering stage is being carried out [9]. This technique recovers the linkages between pieces of legacy source code and the fragments of database schemas used for that pieces. In addition, this knowledge is represented in a *KDM Code Model*. Therefore, in the modernization processes, this new knowledge is essential when a new version of a legacy systems is being developed after the reverse engineering stage. Perhaps, the modernized system does not require all funtionalities of the legacy system. In this case, it is important to be able to identify the fragments of legacy database that are used by the reused pieces of legacy code. As a consequence, the knowledge obtained in the *Data Contextualiztion* could be used to obtain a new database schema that is minimal and fits for the modernized system.

In order to obtain the *code-to-data* linkages this technique considers two knowledge sources: database schemas and the sentences embedded in source code. Thus, the process showed in Figure 1 is divided into three steps: (i) the static analysis of source code (ii) the static analysis of SQL code; and (iii) the model transformations.

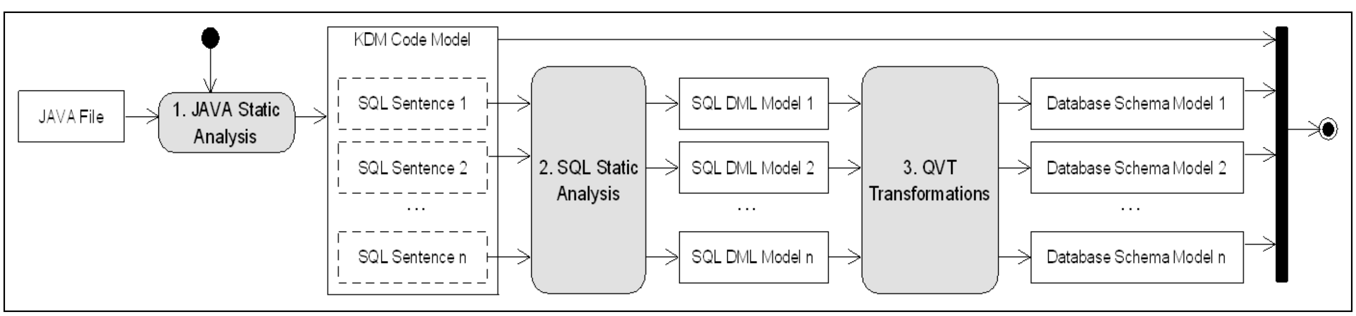

Figure 1. Overview of Data Contextualization: activities, task and artefacts involved in the technique.

A. Static analysis of source code

This activity analyzes the legacy source code in order to obtain a *KDM Code Model*, an abstract conceptual representation of code. This model is represented according to the Code Package of the KDM metamodel. The SQL queries embedded in the source code are also represented in the *KDM Code Model*. Nevertheless, this information is not supported by KDM Code metamodel. For this reason, a specific extension of KDM metamodel according to the extension mechanisms defined in the KDM standard [8] is proposed. Therefore, all generated *KDM Code Models* attach an *ExtensionFamily*.

The *ExtensionFamily* defines three stereotypes: (i) <<SQLStatement>> depicts the SQL code of the queries that is recovered in this activity, (ii) <<SQLModel>> represents the model of each SQL query that is built in the second activity, and (iii) <<DatabaseModel>> represents the model of the a specific database schema fragment obtained from an SQL model in third activity. In addition, each stereotype has a *TagDefinition* to keep the needed information: the code of the query for <<SQLStatement>>, and the path of the model for <<SQLModel>> and <<DatabaseModel>>. Then, when the static parser finds out an SQL query in legacy code, it adds a new *CodeElement* with the stereotype <<SQLStatement>> and it puts the SQL code in an associated *TagValue* element. Thus, the query comes represented into the *KDM code model*.

B. Static analysis of SQL code

After static analysis of source code, there are several points in *KDM Code Model* where the SQL queries were recognized. Thus, this second activity carries out a static analysis of each SQL sentences in the *KDM Code Model* in order to generate several models representing the SQL Sentences. For such an end, a specific metamodel has been developed to represent the embedded SQL sentences. This metamodel represents the Data Manipulation Language (DML) of SQL-92: to model the *Insert*, *Select*, *Update* and *Delete* SQL operations. These operations are generalized in a *SQL Statement* meta-element. A set of *Statements* comprise a DML model. In this point, the *KDM Code Model* knows the *SQL Sentence Models* that it contains in the legacy system, and in turn, the data requirements for each fragment of source code of the legacy system.

C. Model transformations

Finally, in the third activity a *database schema model* is obtained from the *SQL sentence models* by means of a set of QVT transformations according to a database schema metamodel. The database schema metamodel has been developed to represent the database in the *Data Contextualization* process. This metamodel enables the representation of *Tables*, *Constraints* related to these tables, and so on.

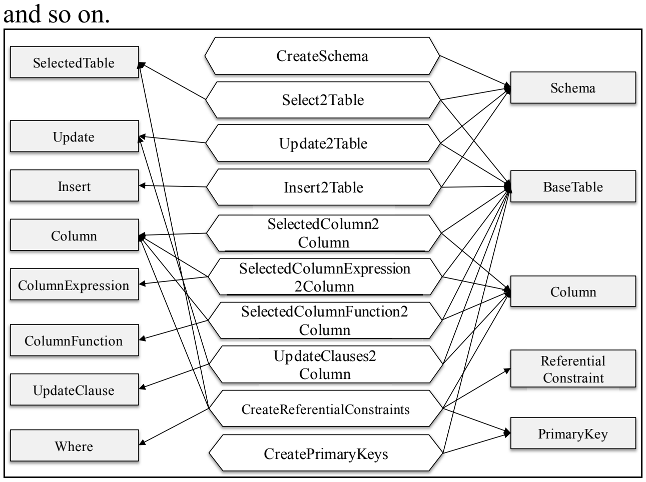

Figure 2. The set of QVT relations

Figure 2 shows the set of QVT relations established between the meta-elements of the *SQL DML Metamodel* (left side) and the meta-elements of the *Database Schema Metamodel* (right side). For instance, the tables that appears in any SQL sentence (*insert*, *select*, *update* or *delete*) as well as in source and target clauses (such as *from*, *set*, *into*, and so on) will be created as *tables* elements in induced database schema. Also, the columns that are selected, added, deleted or updated in SQL sentences will be created in the corresponding tables. In addition, the Select sentences organized in *join* mode suggests potential primary keys and foreign keys in target database schema.

After the QVT transformation, the URL of the obtained database model is put into the *Tag Value* of the *KDM Code Model*. Therefore, the final result is a *Code Model* that has a point for each embedded SQL sentence where it links the associate *SQL Sentence* and *Database Schema Models*. Thus, each piece of source code is related to the database schema fragment that it use.

IV. A TOOL FOR DATA CONTEXTUALIZATION

The Data Contextualization technique is aided by an *ad hoc* tool developed for JAVA-based legacy systems. The tool is structured in three modules corresponding to the three activities of the *Data Contextualization* technique. In order to support the first and second activity, two tool modules to carry out static analysis was developed. Those modules were developed through JavaCC from the EBNF grammar of Java 1.5 and PL/SQL. The first one takes a Java file as input and generates an XMI file as output that represents the *KDM Code model*. The second one takes the previous XMI file and generates several XMI files corresponding to the *SQL sentence models.* Moreover, this module updates the *KDM Code model*, since it puts the URL of the obtained *SQL Sentence models* in each embedded query.

The third module executes the QVT transformation using the *Medini QVT* framework. This module obtain an XMI file for each XMI file correspondig to the *SQL Sentences models*. In addition, the ECORE version of the three proposed metamodels was developed. Indeed, three graphical editors were obtained from them by means of EMF (Eclipse Modelling Framework) tools.

V. CASE STUDY

The case study addresses a modernization project that is currently being carried out. The subject legacy system of this project is the intranet of *Computer Faculty* of *University of Castilla-La Mancha*. This Java-based intranet was developed five years ago by several people. The intranet consists of five well differenced modules: *Main*, *Administration*, *Old Students*, *Management* and *Quality*. The intranet consists of 18.5 KLOC divided into 75 source files. Also, the legacy database schema consists of 140 tables and 7 columns per table on average.

In order to analyze the obtained results, the following research questions are established:

Q1. Are the Database Schema Models complete?

Q2. What is the gain of the Database Schema Models?

Firstly, the *Q1* question is related to the *completeness* of the obtained database schema fragments. A specific schema is complete when: (i) any table has primary key; (ii) there are not tables without columns; and (iii) there are not duplicated elements. Secondly, the *Q2* question takes into account the *minimization* of the database schema. In order to measure the gain between the previous and current size, it uses two variables: the gain related to the number of tables G_T (1) and related to the number of columns in each table G_C (2). In these formulas, T_{LIS} is the number of tables in the legacy database schema and $C_{LIS\{Ti\}}$ represents the number of columns of the table *i* in the legacy database. *T* is the number of tables in the improved database schema and $C_{\{Ti\}}$ is the number of columns of the table *i* in the obtained database.

$$G_T = \frac{T_{LIS} - T}{T_{LIS}} \qquad (1)$$

$$G_{C\{T_i\}} = \frac{C_{LIS\{T_i\}} - C_{\{T_i\}}}{C_{LIS\{T_i\}}} \qquad (2)$$

The case study is focussed particularly on modelling the three kinds of models involved in the *Data Contextualization*. Due to space limitations, this section shows the models obtained through the tool for a specific Java file of the *main* module: *'_Consultar__ Preinscripcion2'*.

Figure 3 (A) shows through the tree model editor the *KDM Code model* obtained after the static analysis of the Java file. Also, it shows two embedded SQL sentences that were discovered. These two points were updated later with the paths of the *SQL Sentence Models* and the *Database Schema Models*. After that, the Data Contextualization tool executes the static analysis of the previous *KDM Code model* and generates two *SQL Sentence Models* for each SQL sentence in the first model.

Figure 3 (B) shows the model related to the second SQL sentence. Finally, the tool executes the QVT relations and generates also two *Database Schema Models* related to the previous models. Figure 3 (C) shows the model related to the second SQL sentence model. In this example, the tables *'MATRICULASCEP'* and *'ALUMNOSCEP'* of a *join select* sentence as well as the columns related to these tables were built in the output model.

TABLE I. THE RESULTS OBTAINED IN THE CASE STUDY.

Module	Legacy System			Database Schema			Gain	
	N. of source files	LOC (mean per file)	N. of Queries (mean per file)	N. of PK	N. of FK	N. of Tables	G_T	G_C (mean per module)
Main	18	323.7	1,8	1	1	9	94%	35%
Administration	8	152.5	1.6	0	0	2	99%	8%
Quality	44	242.0	1.4	0	0	2	99%	29%
Old Students	4	141.8	1.0	0	0	1	99%	80%
Management	1	318.0	1.0	1	1	13	91%	27%
TOTAL	75	18578	1.5	2	2	25	82%	30%

After the execution of the parser and the QVT transformation, a set of output models of the database fragments was obtained. TABLE I summarizes the obtained results; it shows (i) the source files, LOCs and NOQs for each module; (ii) the primary/foreign keys and tables obtained for each obtained database schema; and (iii) the gain obtained with respect to the source database.

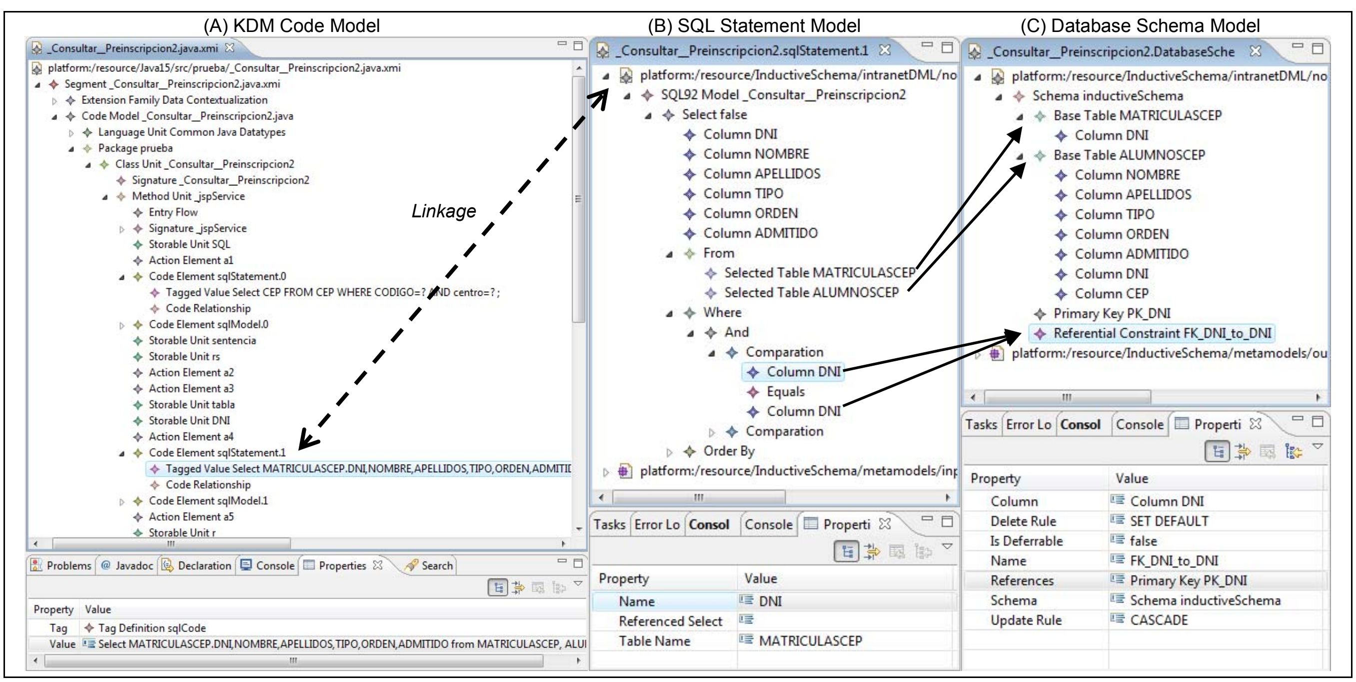

Figure 3. An example of the models involved in the Data Contextualization.

The analysis of results obtained for these models reports several conclusions that should be considered to answer the *Q1* question:

- The tables are usually obtained without primary keys unless a primary key is attached. This problem was solved with a simple matching between the legacy database and the modernized one.
- Achieving tables without columns is not usual, because any column that appears in a SQL statement is normally associated to its table.
- In this case study, since the only QVT-implemented mechanism for inferring foreign keys is the QVT Relation based on the *join select* sentences, the QVT relations do not infer enough foreign keys. Indeed, the source code of intranet has only two join select sentences due to bad design of the legacy database.

In order to response the *Q2* question, the gain of obtained database schema was also assessed. 25 out of 140 tables were recovered (18%) and the G_T value (1) was 82%. Whit respect to the columns, the mean per table of the G_C values (2) was 30%, although in some modules this mean was higher. In this study, the G_C mean is lower than the G_T. However, the total gain related to the size minimization of the new database schema is significant.

VI. Conclusions and Future Work

The *Data Contextualization*, a modernization technique based on KDM, has been proposed in this paper. The objective of this technique is the modernization of legacy source code together with the legacy relational database. For this reason, this proposal recovers the *code-to-data* linkages and obtains three kinds of models according to the ADM approach: (i) The *KDM Code Model,* which represents the inventory of legacy source code. It has also the points that link the *SQL Sentence Models* and *Database Schema Models*. (ii) The *SQL Sentence Model* for modelling a certain SQL query that was embedded in legacy source code. (iii) The *Database Schema Model*, which represents the specific database fragment derived by an *SQL Sentence Model*.

The *Data Contextualization* technique has been validated by means of a case study in a real-life modernization project of a legacy intranet. The case study reports the many advantages and some limitations of the proposed solution. Firstly, the completeness of the database schema model was higher whit respect to table and column elements. Nevertheless, the completeness was lower regarding to the constraint elements. Secondly, the gain of the obtained *Database Schema Models* was important: the size minimization was around the 30% and the 80% for columns and tables respectively.

The work-in-progress focuses on improving the completeness of the output models by means of more patterns related to foreign keys. Furthermore, the future extensions of this research will address the integration of this technique with following stages of the modernization process such as restructuring or forward engineering.

Acknowledgments

This work has been supported by the *FPU Spanish Program*; by the R+D projects funded by *JCCM*: ALTAMIRA (PII2I09-0106-2463), INGENIO (PAC08-0154-9262) and PRALIN (PAC08-0121-1374)*;* and MITOS (TC20091098) funded by the *University of Castilla-La Mancha*.

References

[1] Chikofsky, E., "On the Meeting of Software Architecture and Reverse Engineering", in Working IEEE/IFIP Conference on Software Architecture. 2005, IEEE Computer Society. p. 17-24.

[2] Ghazarian, A., "A Case Study of Source Code Evolution", in 13th European Conference on Software Maintenance and Reengineering (CSMR'09), R. Ferenc, J. Knodel, and A. Winter, Editors. 2009, IEEE Computer Society: Fraunhofer IESE, Kaiserslautern, Germany. p. 159-168.

[3] Lin, L., S.M. Embury, and B.C. Warboys. "Facilitating the Implementation and Evolution of Business Rules". in IEEE International Conference on Software Maintenance. 2005: IEEE Computer Society p. 609-612.

[4] Marinescu, C. "Discovering the Objectual Meaning of Foreign Key Constraints in Enterprise Applications". in 14th Working Conference on Reverse Engineering (WCRE 2007). 2007. Vancouver, BC, Canada: IEEE Computer Society p. 100-109.

[5] Miller, J. and J. Mukerji, "MDA Guide" Version 1.0.1. www.omg.org/docs/omg/03-06-01.pdf 2003: OMG.

[6] OMG. ADM Task Force by OMG. 2007 9/06/2009 [cited 2008 15/06/2009]; Available from: http://www.omg.org/.

[7] OMG, "QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification". http://www.omg.org/spec/QVT/1.0/PDF. 2008, OMG.

[8] OMG, "Architecture-Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM), v1.1." http://www.omg.org/spec/KDM/1.1/PDF/. 2009, OMG. p. 308.

[9] Pérez-Castillo, R., I. García-Rodríguez de Guzmán, O. Ávila-García, and M. Piattini. "MARBLE: Un enfoque ADM para la obtención de Procesos de Negocio". in 6th Taller sobre Desarrollo de Software Dirigido por Modelos (DSDM'09) 2009. San Sebastián, Spain. In press.

[10] Sneed, H.M. "An Incremental Approach to System Replacement and Integration". in Ninth European Conference on Software Maintenance and Reengineering (CSMR 2005). 2005: IEEE Computer Society p. 196-206.

[11] Zou, Y., T.C. Lau, K. Kontogiannis, T. Tong, and R. McKegney, "Model-Driven Business Process Recovery", in Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004). 2004, IEEE Computer Society. p. 224-233.