JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.1554

Software modernization by recovering Web services from legacy databases

Ricardo Pérez-Castillo*[,†], Ignacio García-Rodríguez de Guzmán, Ismael Caballero and Mario Piattini

Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, nº4 13071 Ciudad Real, Spain

ABSTRACT

Databases are considered to be a valuable asset for organizations because they contain all those organizations' persistent pieces of data. Both databases and the information systems that use them undergo erosion as a consequence of uncontrolled maintenance over time. However, when information systems evolve to become modernized versions of them, existing databases must not be discarded because they contain much valuable business knowledge that is not present anywhere else. Some of the software industry's current demands, such as time-to-market developments and the provision of software as services entail additional challenges in the reuse of legacy systems during software modernization. This paper addresses this problem and proposes a reengineering process that follows model-driven development principles to recover Web services from legacy databases. The Web services that are mined manage access to legacy databases without discarding them. Legacy databases can thus be used by modernized information systems in service-oriented environments. The adoption of this process is facilitated by the implementation of a support tool, which is used to conduct an industrial case study involving a real-life legacy database. The study demonstrates that the proposal reduces development efforts and improves the return of investment by extending the lifespan of legacy databases. Copyright © 2012 John Wiley & Sons, Ltd.

Received 25 October 2011; Revised 12 January 2012; Accepted 14 February 2012

KEY WORDS: web services; SOA; software modernization; ADM; relational databases; MDA

1. INTRODUCTION

The ever-growing globalized nature of the world is currently leading to a situation in which organizations are increasingly forced to share more and more data as a basic activity in their daily operations [1]. The heterogeneity of information systems is also growing on a daily basis as a result of the appearance of new technological environments, paradigms, and standards [2, 3]. One consequence of this volatile technological evolution and high level of uncertainty in the software industry is that organizations are involved in a process of continuous renewal of their information and communication technology (ICT) resources to improve, or at least maintain, their competitiveness level through their information systems [4].

These circumstances signify that software engineers and developers need to carry out shorter developments and faster maintenance [5]. This acceleration in the development process involves the reuse, as far as possible, of components and software artifacts that already exist in the organization [6] to attain two main advantages. First, these developments are shorter and cheaper than software developments without reuse. Second, the lifespan of the existing (or legacy) information systems is extended, and the return on investment (ROI) is therefore improved [7].

*Correspondence to: Ricardo Pérez-Castillo, Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, nº4 13071 – Ciudad Real, Spain.
[†]E-mail: ricardo.pdelcastillo@uclm.es

Because the modernization of ICT is a real fact, organizations must make changes to support the evolution of the software implemented in their legacy information systems. In this respect, reengineering has been successfully used over the last two decades to address the necessary evolution of legacy information systems in terms of the migration and reuse of its artifacts [8], such as moving legacy information systems toward environments like the Web while resources such as valuable organizational data are preserved.

Model-driven architecture (MDA) [9] has influenced the software industry in the last few years because this industry is progressively demanding software developments at higher abstraction levels to maximize reuse. MDA treats each system or each piece of systems as models, and establishes model transformations between these models at different abstraction levels, which increases the reusability, formalization, and automation of software developments. Architecture-driven modernization (ADM) [10] also successfully supports those reengineering processes that follow the MDA principles.

Moreover, several software artifacts belonging to a legacy information system (e.g., source code, user interfaces, databases, etc.) have been the subject of reengineering and modernization processes. Nevertheless, databases can possibly be considered as one of the most fundamental artifacts [11] because they contain all of an organization's persistent pieces of data. This is true because data have become the basis of decision-making at operational, tactical, and strategic levels. Consequently, and independently of the various software applications developed, organizations need to keep their data ready to be used within their organizational process.

This paper addresses the aforementioned challenges by proposing an ADM-based process named PRECISO, whose main objective is to expose data stored in legacy databases through a set of Web services. The advantage of this approach is that the data access takes place by means of Web services, thus allowing legacy information systems to be modernized toward SOA environments according to software industry's demands [12]. The ADM-based process consists of the three main stages. First, it recovers functionalities from relational databases by analyzing their schemas. Second, these functionalities are transformed into services. Third, the services obtained are automatically implemented by using Web service technologies. A tool has been implemented to easily support the technique and to facilitate its adoption. This tool has been used to demonstrate the feasibility of the proposed ADM-based process in an industrial case study.

The remainder of this paper is organized as follows. Section 2 introduces the main approaches and standards on which the proposal is based. Section 3 summarizes related work. Section 4 provides a detailed explanation of the ADM-based process. Section 5 presents the most relevant details of the supporting tool. Section 6 provides a case study involving a real-life information system that uses a legacy database. Finally, Section 7 discusses our conclusions and future work.

2. BACKGROUND

The following subsections introduce the three main concepts to provide a better understanding of the proposal: the reengineering approach, the ADM standard, and SOA.

2.1. *Reengineering*

Most companies have existing information systems that are considered to be legacy information systems because the code in these systems was written long ago and may now be technologically obsolete. Software vendors have cultivated a belief that 'anything new is beautiful and that everything old is ugly' and we have become 'victims of a volatile IT industry' [6]. However, legacy software artifacts, like databases or source code, embed much latent knowledge that is not present anywhere else. This valuable knowledge must therefore be preserved when legacy information systems are replaced.

Reengineering has been the most powerful and widely-used mechanism to deal with the knowledge preservation challenge in recent years. Reengineering is 'the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form' [13]. Unfortunately, a 2005 study states that over 50% of reengineering projects fail [14]. This is owing

DOI: 10.1002/smr

to the fact that, in most cases, reengineering usually has two main problems. First, the reengineering of large complex legacy information systems is very difficult to automate [15], and the maintenance costs therefore increase significantly. Second, traditional reengineering processes lack formalization and standardization [16], signifying that different reengineering tools that address specific tasks in the reengineering process cannot be integrated or reused in different reengineering projects.

The software industry is therefore demanding reengineering processes that enable the evolutionary maintenance of legacy systems in an automatic and standardized manner. The typical reengineering concept has therefore shifted to the ADM concept as a solution to these demands.

2.2. *Architecture-driven modernization*

The software modernization paradigm, and particularly ADM as defined by the Object Management Group (OMG), is the concept of modernizing legacy information systems with a focus on all aspects of the current system's architecture and the ability to transform current architectures into target architectures [10]. According to Ref. [17], ADM is the process of understanding and evolving existing software assets, thus restoring the value of existing applications, that is, ADM involves software improvement, interoperability, and migration. The main advantages of using the ADM standard are: (i) the revitalization of legacy information systems, making them more agile; (ii) a reduction in maintenance and development costs; (iii) an extension of the useful life of legacy information systems, while also improving their ROI; and (iv) easy integration with other systems and other environments like SOA.

Architecture-driven modernization advocates carrying out reengineering processes by following the MDA standard [9], which makes it possible to model all the legacy software artifacts as models and establishes model transformations between the different MDA abstraction levels. There are three main kinds of models.

- **Computation independent model (CIM)**, which is a view of the system from the computation independent viewpoint at a high abstraction level. CIM models are sometimes called domain models and play the role of bridging the gap between the domain experts and experts in the system's design and construction.
- **Platform independent model (PIM)**, which is a view of a system from the platform independent viewpoint at an intermediate abstraction level. A PIM has a specific degree of technological independence to be suitable for use with a number of different platforms of a similar type.
- **Platform specific model (PSM)**, which is a view of a system from the platform specific viewpoint at a low abstraction level. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform or technology.

Nevertheless, ADM does not replace traditional reengineering, but improves it. Indeed, the horseshoe reengineering model [16] has been adapted to ADM and is known as the horseshoe modernization model (see Figure 1). This model considers the three main stages of traditional reengineering [9]: (i) reverse engineering (on the left-hand side of the horseshoe), which builds abstract representations of the legacy system at a higher abstraction level; (ii) restructuring (in the curve of the horseshoe), which transforms abstract representations, thus preserving the legacy system's external behavior; and finally (iii) forward engineering (on the right-hand side of the horseshoe), which generates implementations of the target system at a lower abstraction level.

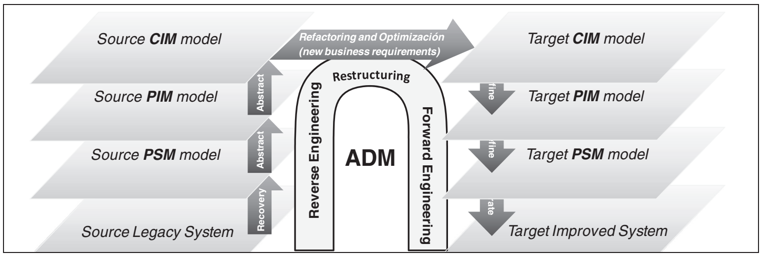


Figure 1. Reengineering process according to ADM approach.

DOI: 10.1002/smr

Architecture-driven modernization solves the formalization problem because it represents all the artifacts involved in the reengineering process as models, and therefore treats them homogenously, that is, as models, meaning that ADM can establish model transformations between them. These transformations are formalized by means of the QVT (query / views / transformations) standard proposed by the OMG [18]. The QVT specification consists of two distinct but related languages: (i) QVT-Operational language, which is procedural in nature, and (ii) QVT-Relations, a declarative language. QVT makes it possible to define deterministic transformations between models at the same abstraction level or at a different level, and the model transformations can consequently be automated. The model-driven development principles also make it possible to reuse the models involved in the ADM projects, because several PIM models can be generated from a CIM model, and various PSM models can in turn be obtained for each PIM model. The automation problem can therefore also be solved as a result of both the automated transformations and the reuse of the models.

To date, model-driven development principles have usually been used in the forward engineering stage. Model-driven generative techniques are used in that stage to obtain source code from different kinds of models, such as UML models. In fact, some researchers consider that model-driven engineering is only applicable to forward engineering. However, model-driven development principles can be applied to the reverse engineering and restructuring stages in an effective manner.

In the reverse engineering stage, the information recovered from software artifacts can be represented in models according to certain metamodels, and restructuring and refactoring techniques can also be applied to models. This stage thus consists of a transformation from the input model (as it is) to obtain a target model (as it will be). Model-based restructuring has some advantages with regard to traditional restructuring: (i) the model-based version allows researchers to define language-independent and platform-independent refactoring techniques; (ii) a restructuring transformation could be implemented as a model in itself, thus allowing the transformation to be reused; (iii) model-based refactoring makes it possible to define generic or domain-specific refactoring techniques in an easy manner; (iv) it improves the feature location because the traceability throughout corresponding models at different abstraction levels is better, etc.

2.3. Service-oriented architecture

Organizations feel increasingly compelled to adopt the new market viewpoint, which is service oriented. This new paradigm has emerged to separate the *possession and ownership* concept from the *use* concept [19]. SOA advocates precisely this approach, that is, offer software as a service [19].

Service-oriented architecture is a means to design, develop, deploy, and manage software systems, and consists of a set of services and service consumers. Services are reusable pieces of source code that represent certain business functionalities. Service consumers can then compose applications or systems by using these services through standard interfaces like WSDL (web service description language).

Service-oriented architecture is consequently the best current option that is available for system integration and the leverage of legacy systems. Indeed, a recent 2010 study developed by the Software Engineering Institute (Carnegie Mellon University, Pittsburgh, Pennsylvania, USA) [12] provides a software engineering research agenda that detects the need for legacy information system modernization towards SOA environments.

3. RELATED WORK

Much work concerning the joint reengineering of applications and databases exists in literature. For example, Reus *et al.* [20] provided a reverse engineering technique with which to transform Procedural Language/Structured Query Language (PL/SQL) textual sources into MDA-ready UML models. Similarly, Pereira and Pinto [21] generated an object-oriented application from the procedures stored in relational databases. The problem with these two works is that they ignore the relational database schema during the reverse engineering stage.

Behm *et al.* [22] transformed relational database schemas into object-oriented schemas and migrated the relational data to the new schema. Hainaut *et al.* [23] proposed a general architecture for data-centered application reverse engineering, and provided a set of CASE tools to deal with this challenge. Finally, they depicted five real-world projects in which the proposal and tools were applied. Polo *et al.* [24]

DOI: 10.1002/smr

proposed a database-driven development method for building applications with a three-tier structure (presentation, business, persistence) from a legacy relational database. The method works as a transformation function that takes the relational schema as its input, producing three sets of classes (which depend on the actual system being reengineered) to represent the final application.

With regard to the reengineering process, the first stage typically focuses on achieving a set of abstract specifications to generate a new system with the new requirements. However, there are many cases in which it is not necessary to create a new database, but rather to wrap the database with an interface to access it, and there is no need to restructure it. These techniques are called ‘wrapping techniques’ and consist of building software components that usually wrap a database management system. For instance, Thiran *et al.* [25] provided a formal framework with which to define schema mappings and to generate as much of the code of the wrappers as possible. McBrien and Pouolovassilis [26] presented a framework that can be used to transform database schemas according to different data models. This work provides a set of transformations with which to automatically migrate or wrap data, queries and updates between semantically equivalent schemas. Moreover, Thiran and Hainaut [27] proposed a generic wrapper architecture to be instantiated for specific data models and software systems. This is achieved by transforming the requests to the subject data model in another target model that is independent of any database management system. Wrapping techniques allow databases to be integrated into new information systems for which they were not initially designed, and the database life cycle can therefore be extended. Canfora *et al.* [28] presented a black-box modernization approach to obtain wrappers that interact with legacy systems through their user interfaces in a noninvasive manner. However, the cornerstone of this approach is not legacy databases.

Despite all these efforts, there is little research in literature concerning the detection of services from databases, that is, obtaining wrappers as services to enable the interaction between modernized information systems and legacy databases. In [6], Sneed proposed a reengineering process to obtain Web services from legacy COBOL applications. In addition, Chung *et al.* [29] provided a method to support legacy information system reengineering, taking Web services as the main building unit. All these works focus on obtaining Web services by analyzing source code and SQL queries embedded in that code, but these proposals ignore database schema’s information. Moreover, these proposals do not incorporate the envision of MDA or ADM approaches to solve the formalization and automation challenges. Indeed, all the work presented in this section usually proposes ad hoc techniques and tools that will hardly ever be reused in other specific situations.

Other proposals attempt to solve the formalization problem by following model-driven development principles. In this respect, Bezivin *et al.* [30] proposed a forward engineering transformation by following the MDA approach to support the generation of Web services. This transformation obtains PSM models, depicting services from different kinds of PIM models such as UML models. Wang *et al.* [31] proposed a framework to support the model-driven reengineering of databases, but this work does not obtain Web services. García-Rodriguez de Guzmán *et al.* [32] provided an approach to support database reengineering to identify Web services. However, although this approach follows the MDA approach, it does not consider certain principles of the new ADM initiative. Nevertheless, this work is the starting point for our research, and the work presented herein is based upon on the idea of relational database schema pattern recognition.

Table I shows a comparison between the current proposal and existing work. The main contributions of this work are: (i) it improves the previous pattern matching technique by adding new patterns to take advantage of foreign key relationships between tables; (ii) this work incorporates the automation of selective publication and the deployment of Web services by following the ADM approach; (iii) it additionally provides a tool that supports and instruments the whole proposal, thus facilitating its adoption in industry; and finally (iv) the proposed approach is validated through a real-life industrial case study.

4. PRECISO — AN ADM-BASED PROCESS

This section describes the ADM-based process named PRECISO, and aims to establish the necessary guidelines to allow the generation of Web services from relational databases. According to the ADM horseshoe model, the proposed process takes a legacy relational database as its input, which is first

DOI: 10.1002/smr

Table I. Comparison of some database reengineering proposals.

Mechanism	Non-model-driven development principles			Model-driven development principles		
Outgoing artifact	Reverse engineering	Restructuring	Forward engineering	Reverse engineering	Restructuring	Forward engineering
New schema	Behm *et al.* [22] Hainaut *et al.* [23]					
New software application	Reus *et al.* [20]; Pereira and Pinto [21] Polo *et al.* [24]			Wang *et al.* [31]		

Wrappers	Thiran *et al.* [25];
	McBrien and Poulovassilis [26];
	Thiran and Hainaut [27];
	Canfora *et al.* [28]					

Services	Sneed [6]; Chung *et al.* [29]			****Our Proposal*** García-Rodriguez de Guzmán *et al.* [32]		Bezivin *et al.* [30]

transformed into a PSM model, according to the SQL-92 metamodel [33], by means of reverse engineering. While other techniques focused on analyzing legacy code to discover integrity constraints or ensure data consistency, this technique is developed to work with relational database schemas, which already contain integrity constraints.

The PSM obtained is then transformed into a PIM model, which raises the abstraction level of the system to the business requirement level, independently of the technology used. The PIM model is depicted by using the UML2 metamodel [34]. The restructuring stage is the right moment at which to introduce new requirements into the PIM model, because it is the starting point from which to generate a new vision of the model. The process is then ready to generate a new PSM model from the PIM, and the forward engineering stage begins here. Because our aim is to generate Web services by allowing the legacy relational database to be accessed, this PSM model must include issues related to Web services technology, and its abstraction level is thus reduced. This PSM model is represented according to the WSDL metamodel.

Despite the fact that PRECISO (the proposed process) is framed in the ADM horseshoe model, it can be seen as a linear process following a sequence of activities that consist, in turn, of a set of fine-grained tasks (see Figure 2). The proposed process considers three main activities: (i) database model recovery, (ii) object model generation, and (iii) Web service generation. These activities are in accordance with the three stages defined by the ADM approach, that is, reverse engineering, restructuring, and forward engineering. The following subsections present all these activities and specify the sequence of tasks involved in each activity. Moreover, Table A.I in Appendix I provides a detailed explanation of the inputs, outputs, and techniques used in the proposed process.

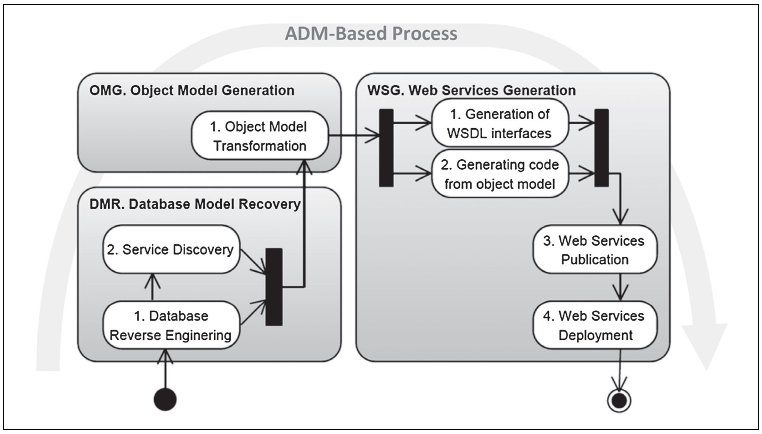

ADM-Based Process
OMG. Object Model Generation
1. Object Model Transformation
WSG. Web Services Generation
1. Generation of WSDL interfaces
2. Generating code from object model
DMR. Database Model Recovery
2. Service Discovery
1. Database Reverse Enginering
3. Web Services Publication
4. Web Services Deployment

Figure 2. Proposed ADM process to generate web services from databases.

In addition, between here and the end of the paper, a small running example is shown to depict how the process works. This example concerns how to recover and develop Web services for a legacy database used by a conference registration application. Figure 3 shows the legacy database schema, which consists of four tables: Conferences, Papers, Authors, and Author-Paper.

4.1. DMR Database model recovery

As previously stated, the first activity aims to create a PSM model of the legacy relational database, which must be migrated to a set of Web services. In addition, potential services must be identified in this activity to ensure that functionalities are not lost. The tasks involved in this activity are detailed in the following paragraphs.

4.1.1. DMR 1 Database reverse engineering. The first task of the modernization process is to use the implemented and working database to abstract a relational model that represents this database. This model is technology-dependent, or in other words, it takes into account the foundations of a specific platform. Some metadata must be recovered as a result. These metadata will be described, for better compatibility and understanding, in terms of the SQL-92 metamodel (see Figure 4), based on Ref. [35].

The information needed to create metamodels of PSMs can be taken from the INFORMATION_SCHEMA [36]. This is a standardized mechanism that is taken from the SQL-92 standard, which identifies the metadata of a particular database through a set of predefined views. These views return database metadata according to a standardized schema. A PSM is consequently built through the information provided by a set of homogeneous queries on the information schema. PSMs concerning database are additionally made persistent by using XMI (XML metadata interchange) [37]. XMI facilitates their safe accurate management, communication, and integration throughout the entire process.

In our example, the implementation of the database model is written using the XMI schema. In this model (see Figure 5), there is an element for each table and for each constraint present in the database schema according to the SQL-92 metamodel (see Figure 4).

4.1.2. DMR 2 Service discovery. A first draft list containing the services that implement the potential functionalities is discovered in parallel during the reverse engineering stage. Bearing in mind that certain patterns are repeatedly used in recovered database schemas, the principal objective of this task is to complete the first draft list with new candidate services by making use of inference mechanisms in these well-known patterns. The discovery of candidate services is based on an already existing technique named 'model driven pattern matching', previously presented in [32], and introduces some new patterns about relational database schemas. PRECISO uses these patterns as a starting point. Moreover, PRECISO improves some of the patterns presented, and adds a new one

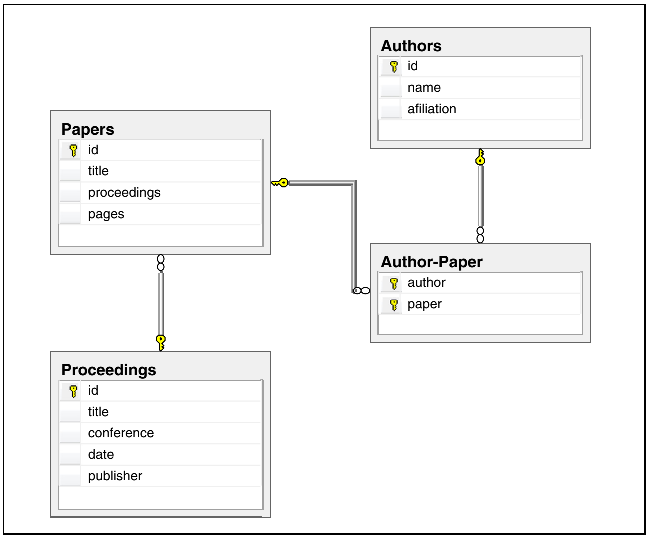


Figure 3. Legacy database schema of conferences system.

DOI: 10.1002/smr

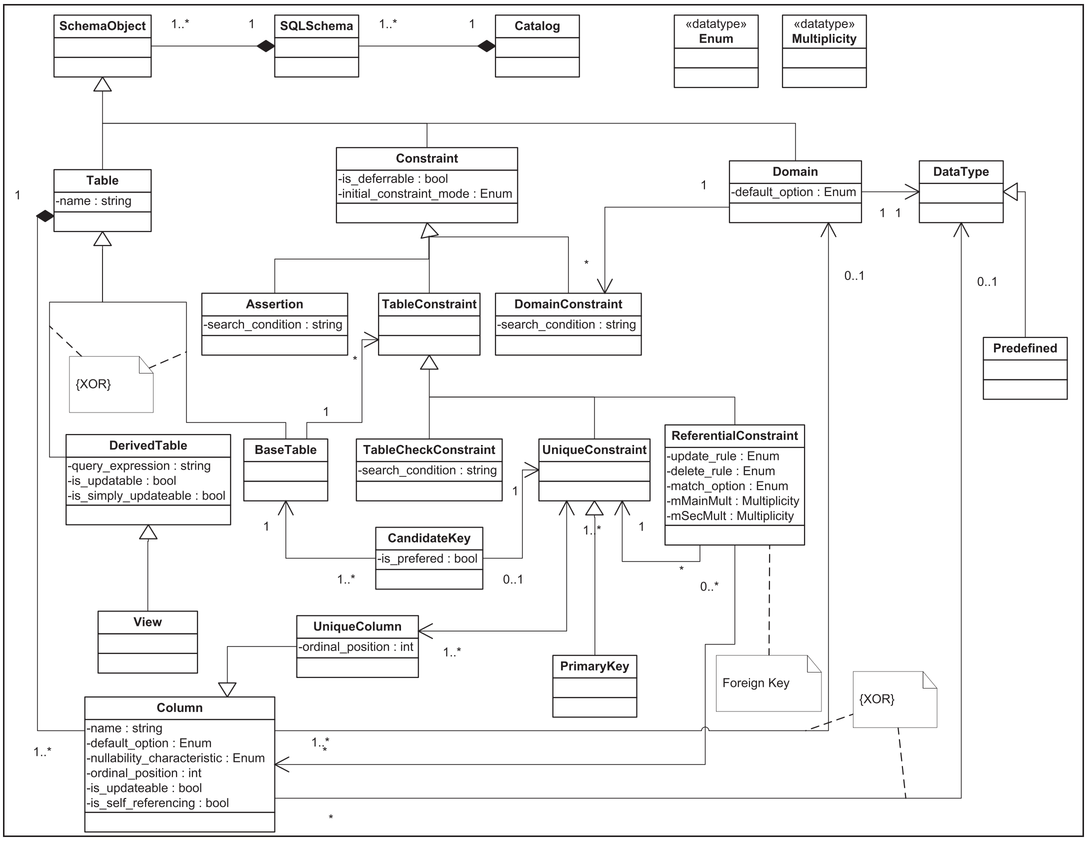
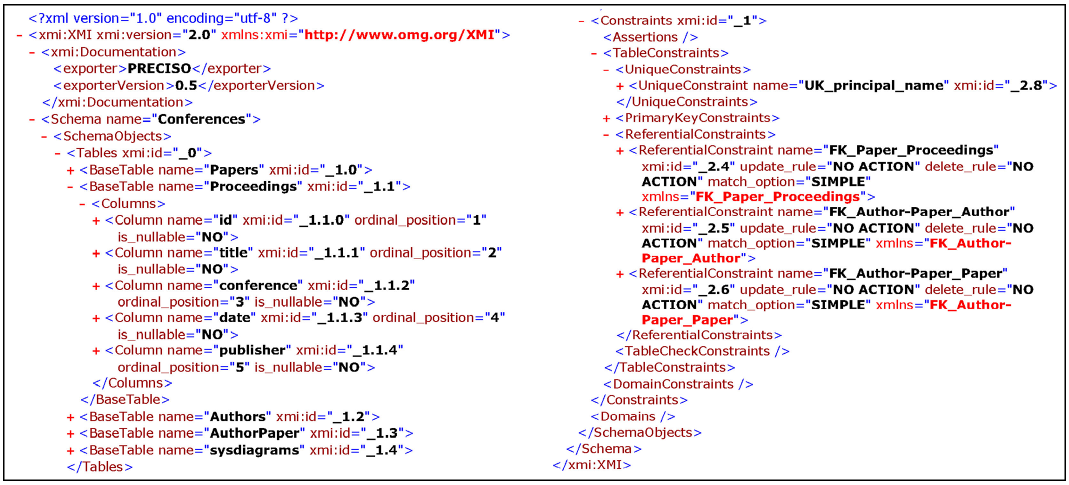


Figure 4. SQL-92 metamodel.



```
  <?xml version="1.0" encoding="utf-8" ?>
- <xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI">
  - <xmi:Documentation>
      <exporter>PRECISO</exporter>
      <exporterVersion>0.5</exporterVersion>
    </xmi:Documentation>
  - <Schema name="Conferences">
    - <SchemaObjects>
      - <Tables xmi:id="_0">
        + <BaseTable name="Papers" xmi:id="_1.0">
        - <BaseTable name="Proceedings" xmi:id="_1.1">
          - <Columns>
            + <Column name="id" xmi:id="_1.1.0" ordinal_position="1"
                is_nullable="NO">
            + <Column name="title" xmi:id="_1.1.1" ordinal_position="2"
                is_nullable="NO">
            + <Column name="conference" xmi:id="_1.1.2"
                ordinal_position="3" is_nullable="NO">
            + <Column name="date" xmi:id="_1.1.3" ordinal_position="4"
                is_nullable="NO">
            + <Column name="publisher" xmi:id="_1.1.4"
                ordinal_position="5" is_nullable="NO">
            </Columns>
          </BaseTable>
        + <BaseTable name="Authors" xmi:id="_1.2">
        + <BaseTable name="AuthorPaper" xmi:id="_1.3">
        + <BaseTable name="sysdiagrams" xmi:id="_1.4">
        </Tables>
      - <Constraints xmi:id="_1">
          <Assertions />
        - <TableConstraints>
          - <UniqueConstraints>
            + <UniqueConstraint name="UK_principal_name" xmi:id="_2.8">
            </UniqueConstraints>
          + <PrimaryKeyConstraints>
          - <ReferentialConstraints>
            + <ReferentialConstraint name="FK_Paper_Proceedings"
                xmi:id="_2.4" update_rule="NO ACTION" delete_rule="NO
                ACTION" match_option="SIMPLE"
                xmlns="FK_Paper_Proceedings">
            + <ReferentialConstraint name="FK_Author-Paper_Author"
                xmi:id="_2.5" update_rule="NO ACTION" delete_rule="NO
                ACTION" match_option="SIMPLE" xmlns="FK_Author-
                Paper_Author">
            + <ReferentialConstraint name="FK_Author-Paper_Paper"
                xmi:id="_2.6" update_rule="NO ACTION" delete_rule="NO
                ACTION" match_option="SIMPLE" xmlns="FK_Author-
                Paper_Paper">
            </ReferentialConstraints>
            <TableCheckConstraints />
          </TableConstraints>
          <DomainConstraints />
        </Constraints>
        <Domains />
      </SchemaObjects>
    </Schema>
  </xmi:XMI>
```

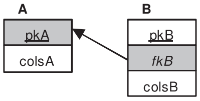
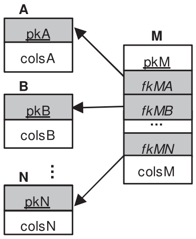
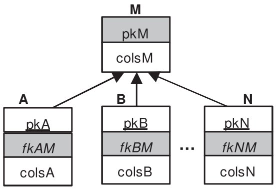
Figure 5. Database model represented by means of its XMI file.

that addresses foreign key relationships between database tables. Table II shows the set of patterns identified in relational database schemas to obtain a set of candidate services. There are simple services that are found in simple database queries and more complex services that take into account both primary key constraints (referenced table and combined table patterns), and foreign key constraints (observed table pattern) (see Table II).

The proposed patterns rely on foreign keys because they represent the relationships between business entities as defined in a company's business processes. This is owing to the fact that

DOI: 10.1002/smr

Table II. Patterns and candidate services which are examined in relational database schema.

Simple Services	Tables	CRUD operations		
		Getters & Setters Operations		
Advanced Services	Views	Queries		
	Patterns in database schema	*Referenced Table*	Select_A_of_B (pkB) Select_B_for_A (pkA)	
		Combined Table	Select_A_for_B (pkB) Select_A_for_B_filtrado (pkB, colsM) Exists_A_related_with_B (pkA, pkB) Select_A_for_B_and_C (pkB, pkC) Select_A_for_B_and_C_filtrado(pkB, pkC, colsM) Exists_A_related_with_B(pkA, pkB, pkC)	 A pkA colsA B pkB colsB N pkN colsN M pkM fkMA fkMB fkMN colsM
		Observed Table	Select_A_for_B (pkB) Select_B_for_A (pkA)	 M pkM colsM A pkA fkAM colsA B pkB fkBM colsB N pkN fkNM colsN

relational models are obtained from entity/relationship (E/R) models, which are previously defined in the design stage. There is consequently a direct relationship between entities and tables, and relationships of E/R models and foreign keys of relational models. Entities (tables) are therefore closely related to the business entities of the company, and relationships between business entities (with a cardinality of ‘1 to *n*’) are related to foreign keys. It is for this reason that the patterns exploit the foreign key information of relational databases.

The technique does not only provide large web services, but also a large set of fine-grained services related to CRUD (Create / Read / Update / Delete) operations and *get* and *set* operations. This ensures the future flexibility of the deployed Web services, which can be composed or choreographed into coarse-grained Web services.

When this task is carried out in our working example, some new services are found by means of the pattern matching technique. More specifically, the ‘combined tables’ pattern (see Table II) is recognized in both the Author-Paper table and the Authors table, and Papers, which contain two respective foreign keys in the Author-Paper table (see database schema in Figure 3). Once the ‘combined tables’ pattern has been recognized, the generation of the predefined services related to this pattern is triggered. Figure 6 shows the services added to the candidate service list after the recognition of the ‘combined tables’ patterns in the aforementioned tables (see Table II and Figure 3).

The output of this task is a set of fine-grained services, corresponding to database queries and mainly obtained from the relationships (foreign and primary keys) between database tables. Although all the services discovered could be published as simple services, they can also be used to build other complex coarse-grained services. Compound Web services are often obtained to support complex functionalities, for example functionalities that are in line with a company’s business operation. This challenge therefore implies that it is first necessary to know the target company’s business processes

```
Author selectAuthors_for_Papers(int id_paper)
Boolean existsAuthors_rel_Papers(int id_author, int id_paper)
Paper selectPapers_for_Authors(int id_author)
Boolean existsPapers_rel_Authors(int id_paper, int id_author)
```

Figure 6. Services obtained from an instance of ‘combined tables’ pattern.

DOI: 10.1002/smr

to adequately manage the necessary orchestration and choreography of such complex Web services. This challenge is not within the scope of the paper, because the technique focuses on the automatic extraction of fine-grained Web services that provide: (i) a wrapper to manage the access to the data stored in legacy databases and (ii) offers a mechanism to integrate legacy databases into SOA environments. Nonetheless, the fine-grained Web services provided by the approach can be used to compose more complex services to provide complex functionalities.

4.2. OMG Object model generation

The principal objective of this second activity is to generate a UML objects model from the information obtained from database recovery. This model of objects represents the PIM model at a higher abstraction level, and will then be the basis used to properly generate Web services in the next activity.

4.2.1. OMG 1 Object model transformation. This activity has a single task, which carries out the PSM → PIM transformation, in which the model obtained of relational database schema is evolved to the object model. The object model is represented according to the UML2 metamodel [34]. Transformations can be formally established by using specific languages to define automatic transformations among models, such as QVT [18]. The transformations can also be carried out manually by writing some pieces of source code for a supporting tool.

The object model obtained in the working example consists of four kinds of objects, one for each table in a database model: Proceedings, Papers, Authors, Author-Paper. These classes will all be part of the so-called Business Layer according to a Three-Tier Architecture [38] (see Figure 7).

4.3. WSG Web service generation

The third activity in the process is the activity that eventually generates and deploys the Web services to manage the initial input legacy database.

4.3.1. WSG 1 Generation of WSDL interfaces. In this task, the object model abstraction is moved to an abstraction level in which the specifics concerning Web services are introduced to obtain a PSM model that supports Web services (see Figure 2). This new PSM model is achieved by merging two input artifacts: the PIM representing the object model and the corresponding model in which

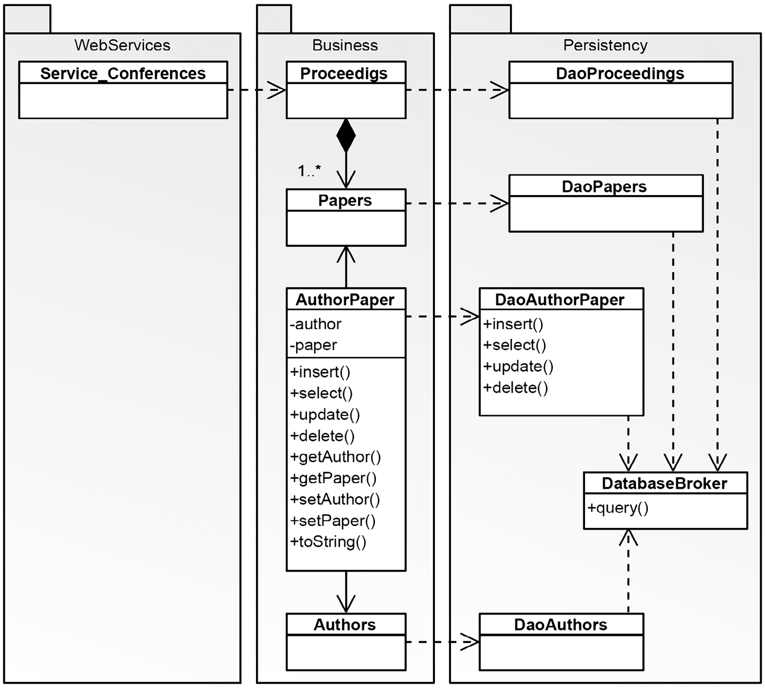


Figure 7. Object model obtained from a conference system in the working example.

DOI: 10.1002/smr

services have been discovered by the pattern matching technique. This task will generate the PSM model written according to the WSDL metamodel [39].

In our working example, we again focus on the discovered instance of the 'combined tables' pattern, and particularly on the first service obtained: 'select Authors_for_Papers(intid_paper)'. Figure 8 shows the representation of a WSDL model segment generated for this Web service after the execution of the DMR 2: Service discovery and OMG 1: Object model transformation tasks.

4.3.2. WSG 2 Generating code from object model. This task aims to generate the pieces of source code to support the object model obtained in previous tasks. This code will be the basis for implementing the Web service architecture.

In our example, the classes represented in Figure 7 are implemented according to a certain programming language that supports Web service technology such as JAVA, C#, and so on. The classes belonging to the Business layer related to the object model are first written to solve possible dependencies. The persistent classes for each business class and the database broker class are then written, taking into account the logical dependences.

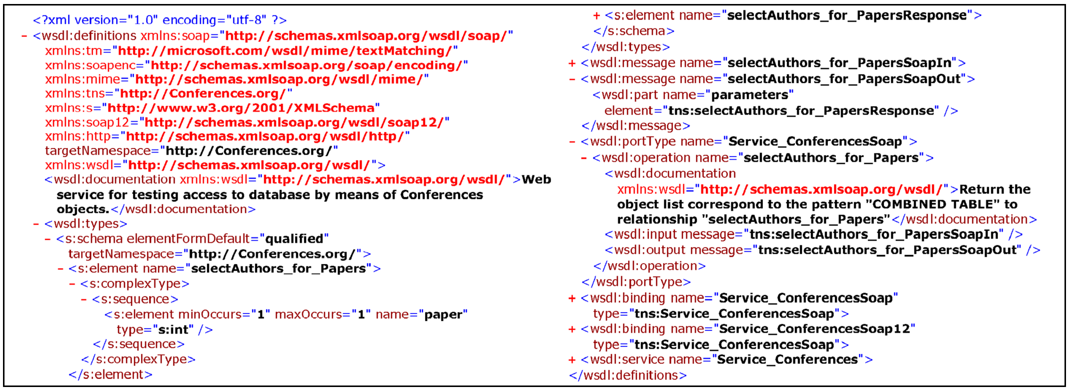
4.3.3. WSG 3 Web services publication. Web services aimed at providing access to the legacy relational database are built using both the source code of the object model and descriptions of WSDL interfaces. The access is carried out by means of a set of published services chosen from among those discovered as candidates during the DMR 2 task.

In the working example, we continue working to enable the selection of services from among all candidate services that were previously discovered. If we return to the services shown in Figure 6, let us consider as an example that analysts might only be interested in services with which to select the rows in the Papers and Authors tables, and they might not be interested in other kinds of services. Analysts will therefore only require these services to be implemented. These two services are published as operations of the Web service class according to a certain programming language that supports the Web service technology (see Figure 7).

4.3.4. WSG 4 Web services deployment. This task is focused on the steps that must be taken to ensure that the Web services chosen will eventually be deployed in a Web server to be used, thus making them fully operational services with which to access the legacy relational database.

This task requires a human-based judgment to select the subset of services to be deployed. This judgment is carried out by system analysts or maintainers with an adequate level of expertise, or could even be done by business experts to deploy Web services that are in line with the company's business processes at a particular moment or in a particular project. However, the services discarded are already useful for future projects in which the requirements might be different.

In our running example, all the artifacts required, which have been obtained by means of the PRECISO process, are automatically deployed in a Web application server and properly configured to permit database responses to external requests.



```
   <?xml version="1.0" encoding="utf-8" ?>
 - <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
     xmlns:tns="http://Conferences.org/"
     xmlns:s="http://www.w3.org/2001/XMLSchema"
     xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
     targetNamespace="http://Conferences.org/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
     <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Web
       service for testing access to database by means of Conferences
       objects.</wsdl:documentation>
   - <wsdl:types>
     - <s:schema elementFormDefault="qualified"
         targetNamespace="http://Conferences.org/">
       - <s:element name="selectAuthors_for_Papers">
         - <s:complexType>
           - <s:sequence>
               <s:element minOccurs="1" maxOccurs="1" name="paper"
                 type="s:int" />
             </s:sequence>
           </s:complexType>
         </s:element>
       + <s:element name="selectAuthors_for_PapersResponse">
       </s:schema>
     </wsdl:types>
   + <wsdl:message name="selectAuthors_for_PapersSoapIn">
   - <wsdl:message name="selectAuthors_for_PapersSoapOut">
       <wsdl:part name="parameters"
         element="tns:selectAuthors_for_PapersResponse" />
     </wsdl:message>
   - <wsdl:portType name="Service_ConferencesSoap">
     - <wsdl:operation name="selectAuthors_for_Papers">
         <wsdl:documentation
           xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Return the
           object list correspond to the pattern "COMBINED TABLE" to
           relationship "selectAuthors_for_Papers"</wsdl:documentation>
         <wsdl:input message="tns:selectAuthors_for_PapersSoapIn" />
         <wsdl:output message="tns:selectAuthors_for_PapersSoapOut" />
       </wsdl:operation>
     </wsdl:portType>
   + <wsdl:binding name="Service_ConferencesSoap"
       type="tns:Service_ConferencesSoap">
   + <wsdl:binding name="Service_ConferencesSoap12"
       type="tns:Service_ConferencesSoap">
   + <wsdl:service name="Service_Conferences">
   </wsdl:definitions>
```

Figure 8. WSDL segment of service 'select authors for papers'.

So far, a general description of the proposed modernization process has been detailed. As previously stated, we have implemented a tool to carry out the application of PRECISO. Details of this tool are provided in the following section.

5. THE SUPPORTING TOOL

PRECISO, the aforementioned ADM-based process, is supported by a stand-alone desktop application (see online [40]). The PRECISO tool leads the analyst through the application of PRECISO, and semi-automatically supports the creation of Web services from legacy relational databases, so that external requests (mainly queries) can be made. As a consequence, and by means of this tool, PRECISO becomes a scalable ADM process that can be used to modernize large legacy databases. Indeed, PRECISO has been applied to a medium/large legacy database in an industrial Web development for a case study, as is shown in Section 6.

PRECISO semi-automates the discovery, publication, and deployment of Web services from legacy relational databases, following the model-driven principles. This means that some human intervention is necessary. In fact, any reverse engineering technique (as our technique is partially) may entail a certain loss of semantics if it is not aided by human intervention. This is owing to the fact that reverse engineering techniques abstract several low-level details (e.g., implementation details).

More details on the design and implementation of the PRECISO tool will be provided below. We have divided the description of the details into two subsections. In Section 4.1, we first summarize some generics in the features of the tool. Second, in Section 4.2, we discuss how some specifics concerning technical details had to be introduced in the development of the PRECISO tool to fully cover all the tasks in the PRECISO process.

5.1. Some generics about PRECISO tool's features

The tool has been designed by bearing in mind its use by developers who wish to continue using existing data in legacy databases, but who wish to migrate the application to other new platforms, or even to include new features in existing ones. The main benefit of using the PRECISO tool is the reduction of development times when dealing with the persistence layer of an Model/View/Controller (MVC) structured application. This has become a reality because of the automatic generation of Web services.

As previously stated, the tool addresses issues such as remote database connection, connections to databases from different manufacturers, the project-based graphical display of generated models, testing, reporting, and so on. The proposed architecture, which takes into account the aforementioned challenges, is shown in Figure 9

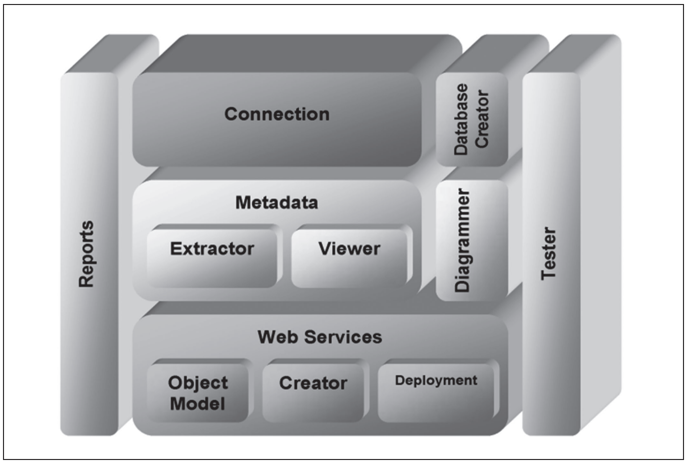


Figure 9. Architecture of the developed tool.

DOI: 10.1002/smr

5.1.1. High coverage of the PRECISO architecture-driven modernization process. The PRECISO tool has been developed to fully support the PRECISO process. The generation of Web services from relational databases is therefore widely supported throughout all the stages involved in the modernization process: metadata extraction, model generation, model visualization, object model transformation, editing and publication of Web Services, deployment, testing, reporting, and so on.

5.1.2. Working in a 'project-based' way. The PRECISO tool manages each instance of the PRECISO process as a minimal full-sense working unit under the name of 'project' (Figures 10 and 11). This has two advantages: on the one hand, the project enables the generation of Web services to be resumed

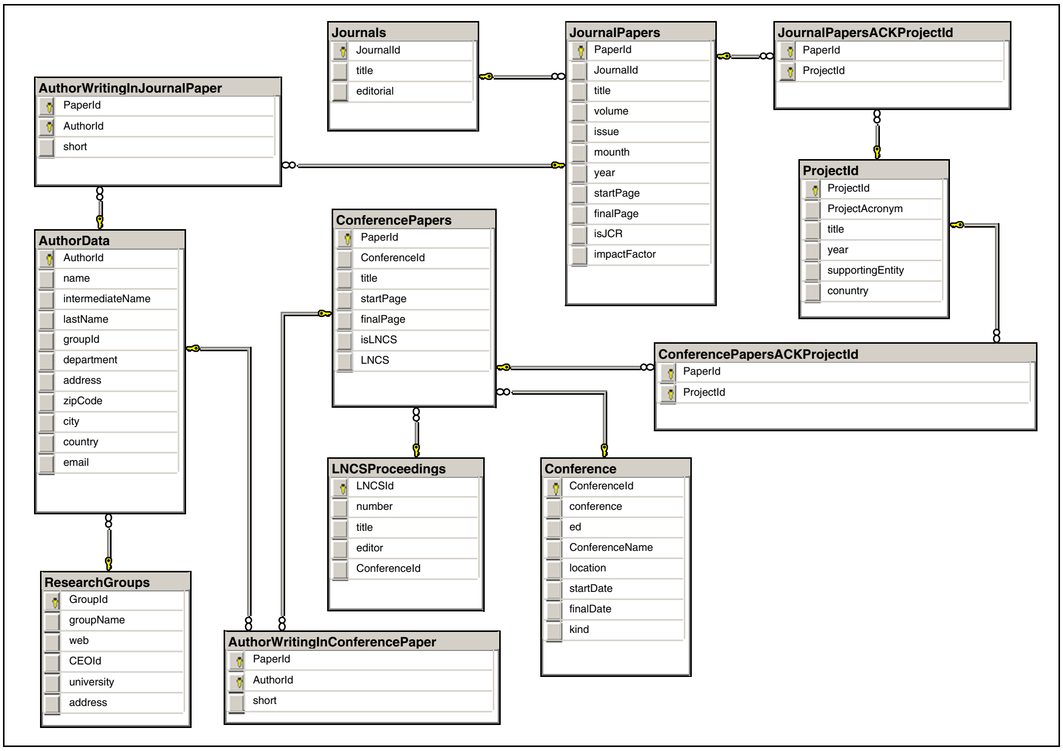
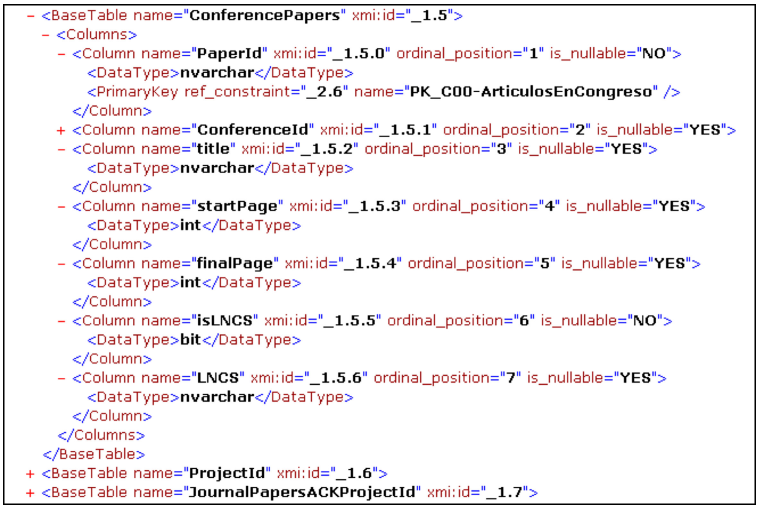


Figure 10. Relational schema of legacy database of CI.

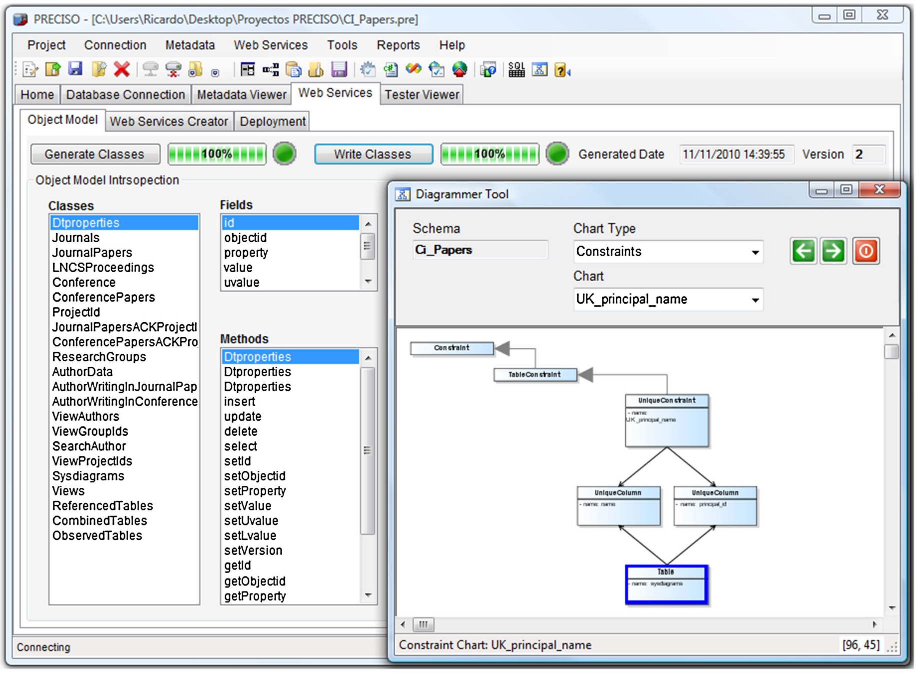


```
- <BaseTable name="ConferencePapers" xmi:id="_1.5">
  - <Columns>
    - <Column name="PaperId" xmi:id="_1.5.0" ordinal_position="1" is_nullable="NO">
        <DataType>nvarchar</DataType>
        <PrimaryKey ref_constraint="_2.6" name="PK_C00-ArticulosEnCongreso" />
      </Column>
    + <Column name="ConferenceId" xmi:id="_1.5.1" ordinal_position="2" is_nullable="YES">
    - <Column name="title" xmi:id="_1.5.2" ordinal_position="3" is_nullable="YES">
        <DataType>nvarchar</DataType>
      </Column>
    - <Column name="startPage" xmi:id="_1.5.3" ordinal_position="4" is_nullable="YES">
        <DataType>int</DataType>
      </Column>
    - <Column name="finalPage" xmi:id="_1.5.4" ordinal_position="5" is_nullable="YES">
        <DataType>int</DataType>
      </Column>
    - <Column name="isLNCS" xmi:id="_1.5.5" ordinal_position="6" is_nullable="NO">
        <DataType>bit</DataType>
      </Column>
    - <Column name="LNCS" xmi:id="_1.5.6" ordinal_position="7" is_nullable="YES">
        <DataType>nvarchar</DataType>
      </Column>
    </Columns>
  </BaseTable>
+ <BaseTable name="ProjectId" xmi:id="_1.6">
+ <BaseTable name="JournalPapersACKProjectId" xmi:id="_1.7">
```

Figure 11. XMI code of table 'conference papers'.

DOI: 10.1002/smr

from a previous point that has been saved. On the other hand, the project file can be migrated to another computer. A project is composed of a package of information concerning: the database to be modernized, the object model transformations, publication and deployment of services, and so on. Typical operations can be carried out in the project file: creation, open, save, and so on. Throughout Section 6 (corresponding to the case study presented in this work), some snapshots of the tool's user intertool are provided and explained (see Figures 12 and 13).



Generated Date 11/11/2010 14:39:55 Version 2

Object Model Intrspection

Classes: Dtproperties, Journals, JournalPapers, LNCSProceedings, Conference, ConferencePapers, ProjectId, JournalPapersACKProjectI, ConferencePapersACKPro, ResearchGroups, AuthorData, AuthorWritingInJournalPap, AuthorWritingInConference, ViewAuthors, ViewGroupIds, SearchAuthor, ViewProjectIds, Sysdiagrams, Views, ReferencedTables, CombinedTables, ObservedTables

Fields: id, objectid, property, value, uvalue

Methods: Dtproperties, Dtproperties, Dtproperties, insert, update, delete, select, setId, setObjectid, setProperty, setValue, setUvalue, setLvalue, setVersion, getId, getObjectid, getProperty

Schema: CI_Papers

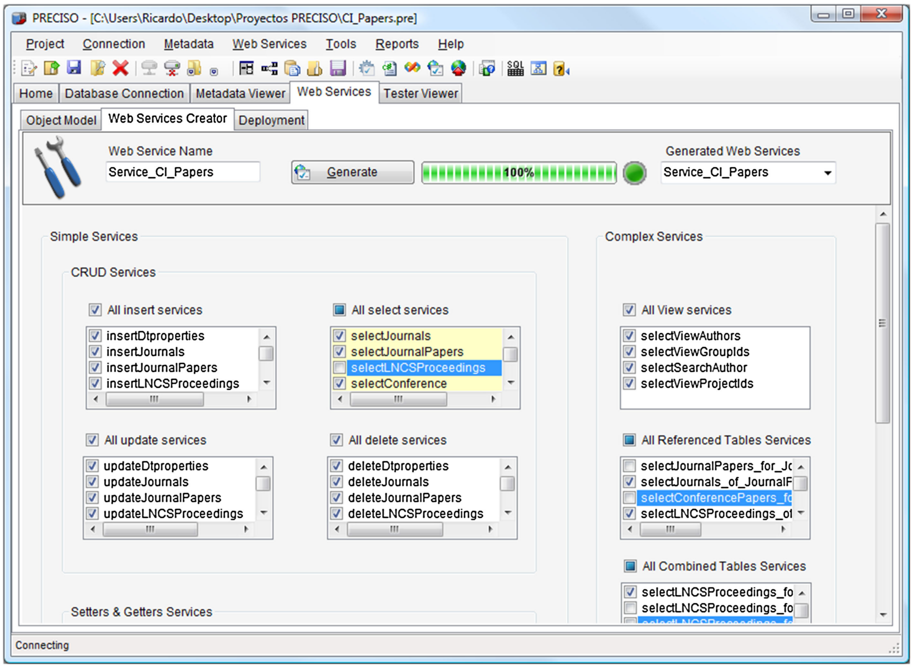
Chart Type: Constraints

Chart: UK_principal_name

Constraint Chart: UK_principal_name [96, 45]

Connecting

Figure 12. Class model obtained from the database model.



Web Service Name: Service_CI_Papers

Generated Web Services: Service_CI_Papers

Simple Services

CRUD Services

☑ All insert services
- ☑ insertDtproperties
- ☑ insertJournals
- ☑ insertJournalPapers
- ☑ insertLNCSProceedings

■ All select services
- ☑ selectJournals
- ☑ selectJournalPapers
- □ selectLNCSProceedings
- ☑ selectConference

☑ All update services
- ☑ updateDtproperties
- ☑ updateJournals
- ☑ updateJournalPapers
- ☑ updateLNCSProceedings

☑ All delete services
- ☑ deleteDtproperties
- ☑ deleteJournals
- ☑ deleteJournalPapers
- ☑ deleteLNCSProceedings

Setters & Getters Services

Complex Services

☑ All View services
- ☑ selectViewAuthors
- ☑ selectViewGroupIds
- ☑ selectSearchAuthor
- ☑ selectViewProjectIds

■ All Referenced Tables Services
- □ selectJournalPapers_for_Jc
- ☑ selectJournals_of_JournalF
- □ selectConferencePapers_fc
- ☑ selectLNCSProceedings_of

■ All Combined Tables Services
- ☑ selectLNCSProceedings_fo
- □ selectLNCSProceedings_fo

Connecting

Figure 13. Selective deployment of web services through PRECISO tool.

DOI: 10.1002/smr

5.1.3. Partitioned and ordered process generation. In the PRECISO tool, each activity of the process is associated with an individual module. The main reason for doing this is that each module can be executed in an independent manner, both whenever it is suitable to do so, and wherever it can be done, that is, by different members of staff on several computers. Figure A.1 (Appendix II) depicts the manner of working as a partitioned process implemented by the PRECISO tool. The representation is shown by means of a UML state machine diagram.

5.1.4. Alignment to widely-used standards and extensible to others. To ensure that they would be adaptable to any of the possible environments, we decided to align the artifacts generated and used by the PRECISO tool to as many technical 'de jure' standards as possible. In this respect, the PRECISO Tool uses standards such as SQL-92 to inspect the database; XML and XMI to represent the metadata [41] and to represent models; WSDL [39] and SOAP (Simple Object Access Protocol) to support Web services technology [42], and so on. These standards allow the artifacts generated by the PRECISO tool to be integrated into other modernization and case tools on the market, which supposes an important benefit in the saving of both time and resources when several development platforms must be used

To obtain a better performance and satisfy other standards, we contemplated the need to allow the PRECISO tool to be sufficiently extensible to manage artifacts according to any other standards that are not as widely usable as those for which we have implemented support.

5.1.5. Generation of software rather than simply source code. One of the most challenging issues that we wished to confront when developing the PRECISO tool was to deploy running software rather than simply generating source code for Web services. As software engineers, we are conscious of the importance of properly documenting the generated artifacts. We therefore decided that the tool should not only accomplish the requested pieces of running software, but should also automatically accomplish a set of deliverables that complement and improve the generated source code, such as class documentation, reports, model diagrams, database scripts, and so on.

5.2. Some specifics regarding the PRECISO tool's features

As previously mentioned, the PRECISO tool covers each task in the PRECISO process. However, to successfully implement this coverage, some decisions on technical details had to be made during the development. In this subsection we analyze, for each of the PRECISO process tasks, the specifics on which we based our design and implementation decisions to develop certain features.

5.2.1. Specifics regarding DMR Database model recovery. The functionalities needed to recover database models from existing legacy relational databases have been implemented in the PRECISO tool, along with certain strategies and heuristics to identify candidate services. Because one of our aims was to align the design to as many standards as possible, we decided to focus on which were the most important ones. For instance, for the **DMR1 Database reverse engineering** activity, we decided that models derived from databases (PSMs) would be represented according to the SQL-92 metamodel [35]. This decision was made because the SQL-92 standard [33] is widely used in the software industry [43, 44]. However, the PSM models can be also recovered from any other database using other standards such as SQL-86, hierarchical databases, databases based on COBOL, and so on. Because we required the tool to be extensible, it was only necessary to add the corresponding metamodels to properly represent these models.

On the other hand, we found it necessary to describe how we proceeded so that readers could better understand the path we had taken. We are now therefore able to, for example, provide some details on the **DMR 2 Service discovery** activity, which examined the database model obtained to identify some candidate services. The easiest, most reliable, and quickest way to do this is to infer them from the discovery of certain structures based on pattern matching techniques [32]. We therefore decided to use these standard techniques (see Table II for details on both the search patterns and the services that can be derived from each pattern). It is worth highlighting that there are simple services that involve only a single table. These services are directly obtained from the database scheme and matched to CRUD operations along with *getters & setters* methods to handle various columns in each table. Furthermore, advanced services involve several tables of schema (Table II). In this case,

DOI: 10.1002/smr

services may be directly obtained from views or, on the contrary, services could be obtained from the following patterns that were identified in the relational database scheme: (1) referenced table, when there is a foreign key among two tables; (2) combined table, when there are two or more foreign keys from one table to another; and (3) observed table. Unlike the second table, this pattern searches two or more foreign keys in the same table.

5.2.2. Specifics on OMG Object model generation. The aim of the OMG task is object model generation. This task corresponds with the module for which some specifics are to be detailed.

In the **OMG1 Object model transformation** task, the tool must address a PSM → PIM transformation. Because we consider that SQL-92 represents database models, this is our source metamodel in the transformation. We believed that the target metamodel should be a UML2 model because it is the standard in software development. It was also necessary to define how to implement the transformation by choosing a strategy from the possible options: using QVT or directly implementing it by means of code within the tool. After analyzing the suitability of each option, we decided to use and consequently implement the algorithm depicted in [30], which executes the following transformations: a table → a class, a column → an attribute, a foreign key → association, and so on. Moreover, SQL-92 data types are mapped onto data types of the generic programming language, which is used to generate source code.

5.2.3. Specifics on WSG Web services generation. Finally, because we wished the tool to generate running software rather than source code, we had to implement the publication and deployment of Web services. This requirement involved an analysis whose specific details are presented in the following paragraphs.

The **WSG1 Generation of WSDL interfaces** task takes as input both the previously obtained object model and the services discovered. These two artifacts are the basis used to generate the Web service descriptions, using WSDL interfaces as its output.

With regard to the task named **WSG 2 Generating code of object model**, and to deploy executable Web services, the tool must write files containing the source code of the object model in a persistent store. The source code must be written in any language with Web services technology support. In this tool, we decided to use *C#.Net*. In addition, the PRECISO tool supports the edition of the files generated. Anyway, the programming language used to implement the object model supporting the web service interface is independent of the proposed technique and could be extended by providing different modules to generate source code in different programming languages.

For the **WSG 3 Web services publication** task, the tool is prepared to offer users the functionality of selecting services from among those services discovered. Users can select services that will be included in the Web service, which will eventually be generated. Several reasons, including those related to security, signify that it is not advisable to make the access to the entire database public by means of certain services that will not be used in the future. Moreover, because the PRECISO tool can generate several Web services with different services, it would be possible to generate several Web services with different sets of services to provide different partial viewpoints of the database.

To implement the **WSG 4 Web services deployment** task, the tool must set up the Web services built to enable them to be executed on the Web, so that they can be properly used. A particular application server with the capability of deploying Web services should be selected for this. Of all the existing commercial software, we decided that the PRECISO tool would be prepared to interact with Microsoft Internet Information Server 6 (IIS6), because it is only integrated in the .Net platform used during the development and writes the software corresponding to the object models. The PRECISO tool had to be able to manage the various means of deployment supported by IIS6: (1) source code files are copied in the correct location of the application server and (2) this source code is stated as a Web directory that is accessible on the Web.

Finally, because of the fact that the implementation of the tools follows the model-driven development principles, any implementation-specific adjustments will be easier to make than with other tools. Different abstraction levels and their respective models at each level are consequently less coupled than tools that do not follow model-driven principles. For example, if a new SOA platform is, in the future, the target technologies for which Web services are created, then only some

DOI: 10.1002/smr

adjustments are necessary. First, the PIM metamodel would be the same, and second a new transformation between the PIM metamodel and the new PSM metamodel would be necessary.

6. CASE STUDY

This section presents a case study to demonstrate the feasibility of PRECISO, the proposed ADM-based process, by applying it to an industrial project. This project addresses the modernization of a legacy database to integrate it into a new website. The case study was carried out by following the protocol for planning, conducting, and reporting case studies proposed by Runeson and Host [45]. The following sections present the details of the main stages defined in the formal protocol: background; design; case selection; case study procedure; data collection, analysis and interpretation; and validity evaluation.

6.1. Background

The case study consists of a project that was jointly carried out by the University of Castilla-La Mancha and Indra Software Labs (Ciudad Real, Spain) (a multinational software company) in the context of the 'CATEDRA INDRA', an R&D centre (located in Spain) that carries out research projects in close cooperation between industry and the university. This research centre is supported by the 'University of Castilla-La Mancha' (UCLM) and 'INDRA Software Labs at Ciudad Real'.

CATEDRA INDRA (hereafter CI) needed to develop its corporate portal[1] to support all the information produced from the cooperation between industry and the university. This site is directed towards academics, researchers, teachers, PhD candidates, and PhD students. The site contains information about conferences, lectures, courses, grants offered, events, awards, papers, journals, and so on.

The CI portal has been built using a standard Web architecture based on the Microsoft .NET platform. On the one hand, it has used Microsoft Content Management Server 2002 (MCMS) as a content management system (under the RDBMS Microsoft SQL Server 2000), and on the other hand, Active Server Pages (ASP) has been used for the presentation layer. Finally, the whole application (that is, the MCMS and the Web application) has been deployed through Microsoft Internet Information Server 6 (IIS6) in a MS Windows Server 2003 host.

Because of space limitations, the case study focuses on a certain submodule of the CI portal. This module deals with the tasks needed to manage the research papers produced by CI. This module must be able to search research papers according to different criteria, and it must also be able to add new paper information, and modify or delete existing papers. Until that moment, all these activities had been carried out manually supported by the tables of a legacy relational database. The new CI portal will be based on an existing database provided by the CI. This database, which was created many years ago, stores a considerable amount of information from existing publications by people involved in the CI. This information is not managed by any application and it is therefore possible to find the suitable preconditions for implementing a modernization process through the tool. The CI's database is thus considered to be a legacy system. The case study applies PRECISO, the proposed ADM-based process, to recover a set of Web services as a wrapper to integrate the legacy database into the entire CI portal.

6.2. Design

The development staff responsible for this project was interviewed to understand the information needs of the module of the CI portal. The interviews were analyzed and this analysis reported the information needs according to established user requirements. This meaningful information would help in the usage of the tool to obtain the set of correct Web services required to feed the Web layer by means of the appropriate information.

[1] http://catedraindra.uclm.es

DOI: 10.1002/smr

In brief, the problem is the following: it implements a module to query information about the publications of the CI researchers. It must therefore store all the information from conferences and journals, data concerning authors, R&D projects financed, and so on. Moreover, this module must contain a search engine to set up filters to carry out customized searches according to different search criteria such as conference, journal, international/national, whether a conference is a Lectures Notes in Computer Sciences (LNCS), whether a journal is indexed, authors, and many other criteria.

The case study design consists of a single case, that is, it focuses on a single legacy database. This study can therefore be considered as an holistic case study according to the classification proposed by Yin [46]. The object of study is PRECISO, the proposed ADM-based process, and the purpose of the study is the evaluation of specific properties of PRECISO related to effectiveness and efficiency. Bearing in mind the object and purpose of the study, the main research question (hereafter, MQ) addressed by the study is the following: Can PRECISO obtain Web services from legacy databases? To answer the main research question, two additional subquestions are proposed to evaluate whether Web services can be obtained in an effective and efficient manner (Table III). The additional research question, AQ1, is stated to evaluate whether the Web services obtained are able to manage all the data stored in the legacy database. That is, AQ1 aims to evaluate the effectiveness of the procedure. Moreover, the additional research question AQ2 (Table III) evaluates whether PRECISO can be used in real-life modernization projects to obtain Web services with a moderate effort, that is, whether the proposed ADM-based process can obtain Web services efficiently and is consequently scalable to different kinds and sizes of databases.

The study considers some measures as dependent variables, which facilitate a quantitative analysis that can be used to provide an answer to the research question. On the one hand, to answer the AQ1 question related to effectiveness, all the different kinds of services (regarding the pattern that is applied in each case) are measured for each legacy database table (e.g., CRUD, setter/getter/show, referenced table, combined table, observed table). Another measure is the *Usage ratio* (1) of services, that is, the percentage of services that have already been published and deployed from the entire set of candidate services. For this measure, it is important to know the real level of usage of all of the services discovered. On the other hand, to provide an answer to the AQ2 question related to efficiency, the development effort is estimated when (i) PRECISO is used to integrate the legacy database by means of the Web services and (ii) when the integration of data management has to be developed from scratch. The effort involved in a new development from scratch (i.e., throw away the legacy database) is estimated in hours by taking into account the historical data of similar development projects carried out by the company. The effort of database wrapping development through PRECISO is accurately measured in hours after carrying out the case study. The study uses the *Effort gain* measure (2) to discover the difference between these two kinds of efforts. *Effort gain* is measured as the difference between the planned effort and actual effort for a particular development activity divided by the difference of both total efforts.

$$Usage\ ratio = \frac{Published\ services}{Candidate\ services} \tag{1}$$

$$Effort\ gain_i = \frac{E_{i_{\mathrm{PLANNED}}} - E_{i_{\mathrm{ACTUAL}}}}{E\mathrm{TOTAL}_{\mathrm{PLANNED}} - E\mathrm{TOTAL}_{\mathrm{ACTUAL}}} \tag{2}$$

Table III. Research questions of the case study.

Id	Research question
MQ	Can PRECISO obtain web services from legacy databases?
AQ1	Can PRECISO accurately obtain web services to manage all the stored data?
AQ2	Can PRECISO efficiently obtain web services with a moderate effort?

DOI: 10.1002/smr

6.3. *Case selection*

The case selection is a key stage in the case study planning, which aims to select a good and suitable case to be studied. The study establishes four criteria (C1 to C4) to select the most appropriate database. C1 guarantees that the legacy database selected is a database that stores the organization or company's business data. This criterion discards, for example, databases supporting information of embedded systems or real-time systems. C2 ensures that the legacy database will be a real-life database that is deployed in a production environment. C3 ensures that the selected database really is a legacy database. To evaluate this criterion the time in production is not used, because it is not a good measure. Instead of production time, C3 considers the amount of modifications in the database that alter the schema database. Finally, C4 guarantees that the database is a relational database, because both the proposed ADM-based process and the supporting tool were especially developed for relational databases. Moreover, C4 ensures that the relational schema contains all the integrity constraints. The study could consider a legacy database without integrity constraints, and previously apply an algorithm to discover the implicit constrains in code. However, these techniques are not the main objective of this study and may introduce side effects in the technique under study.

After evaluating several available databases according to the aforementioned criteria, the legacy database of the CI was selected to be studied. The CI's database stores all the information related to different kinds of papers, authors, conferences, and so on, and thus meets the C1 criterion. The first release of this database was moved to the production stage two years ago. It is used manually by CI's staff, and therefore also meets the C2 criterion. During that time, the CI staff in charge of managing the papers made four medium modifications (versions 1.1, 1.2, 1.3, and 1.4), a large modification (version 2.0), and three more medium modifications (versions 2.1, 2.2, and 2.3). This ensures compliance with the C3 criterion. From a technological point of view, the CI's database was implemented by using SQL Server 2000 (Microsoft) as the Database Management System (DBMS), that is, a relational database technology. The C4 criterion is therefore also met.

The schema of the selected legacy database is shown in Figure 10. The main tables are 'Conference Papers' and 'Journal Papers'. These main tables store papers produced for conferences and papers produced for journals, respectively. These tables are then associated with other tables. These other tables specify, among other things, data concerning the papers' authors ('Author Data'), the conference data ('Conference'), data about journals ('Journals'), and so forth.

6.4. *Execution procedure*

After the design and case selection of the study, the execution procedure of the study also had to be planned. The execution was aided by the tool developed to support the procedure. The case study procedure defines the following steps. The tool first establishes a connection with the database of the CI. It subsequently generates an XMI file containing all the metadata concerning the database studied. The piece of XMI code, corresponding to the 'Conference Papers' table, is shown as an example in Figure 11.

In the second step, the object model needed to support the future Web services was created. The tool generated necessary classes depending on the metadata obtained in the previous stage. The tool also wrote executable classes (in this case, using *C#* language). Figure 12 shows the class model of both the domain and persistence tier corresponding to the database model obtained.

Third, the PRECISO tool achieved the executable Web services by using the class model created in the previous step. Furthermore, the tool allows the selective publication of services. It was therefore sufficient to create only some public services. These exposed only the parts of the database that were necessary for the development of the project (see Figure 13). The tool then transformed Web services into operational Web services and carried out their deployment in a Web application server. Suitable Web services with which to provide the required functions for handling the database were therefore obtained in accordance with the previously imposed information needs. Finally, the set of

DOI: 10.1002/smr

recovered Web services were integrated into the CI portal, and Web services could thus be invoked from Web functions in the user interface layer.

6.5. Data collection

It was necessary to define which data needed to be collected, and the data sources, before starting the execution of the case study. First, Table IV collects data related to the legacy database, along with the candidate services discovered through the tool. Table IV shows (i) the name of each table; (ii) the number of columns in each table; (iii) the number of candidate services grouped by each kind of service (i.e., CRUD services; services supporting setters, getters and shows operations; services according the proposed patterns such as referenced table, combined table and observed table); and finally (iv) the total number of candidate services for each table.

Second, Table V collects data related to the final services published and deployed to use in the CI portal. Table V shows (i) all the kinds of services; (ii) the total number of services for each type;

Table IV. Candidate services obtained for each table.

		Kind of service					
Table/View	#Columns	CRUD	Setter/getter/ show	Referenced table	Combined table	Observed table	Total
Journals	3	4	7	0	0	0	11
Journal papers	11	4	23	2	5	0	34
LNCS proceedings	5	4	11	0	6	0	21
Conference	8	4	17	0	6	0	27
Conference papers	7	4	15	4	5	0	28
Project Id	6	4	13	0	4	0	21
Journal papers ACK project Id	2	4	5	4	0	2	15
Conference papers ACK project Id	2	4	5	4	0	2	15
Research groups	6	4	13	0	0	0	17
Authordata	11	4	23	2	6	0	35
Author writing in journal paper	3	4	7	4	0	2	17
Author writing in conference paper	3	4	7	4	0	2	17
View authors	11	0	1	0	0	0	1
View group Ids	6	0	1	0	0	0	1
Search author	9	0	1	0	0	0	1
View project Ids	6	0	1	0	0	0	1
Total	99	48	150	24	32	8	262

Table V. Published services in CI portal.

Kind of service		Candidate services	Published services	Usage ratio	
CRUD	Insert	12	11	85.0%	93.8%
	Update	12	11	85.0%	
	Delete	12	11	85.0%	
	Select	12	12	92.0%	
Setters/getters/shows	Setters	67	0	0.0%	2.7%
	Getters	67	0	0.0%	
	Show	12	0	0.0%	
	Views	4	4	100.0%	
Referenced tables		24	12	50.0%	
Combined tables		32	8	25.0%	
Observed tables		8	4	50.0%	
Total		262	73	27.9%	

(iii) the total number of services published in the CI portal; and finally (iv) the exploitation performance of each kind of service, that is, the percentage of published services.

Table VI presents (i) the six activities planned in the software development plan; (ii) the effort that was previously estimated; and (iii) the effort that was eventually needed to carry out each activity. The effort was measured in the work hours of a single developer.

6.6. Analysis and interpretation

The MQ question could not be answered until the AQ1 and AQ2 questions had been answered (Table III). The tool carried out a selective publication and deployment of the Web services generated from the legacy database. The total number of candidate services was 262 (Table V). The Web services needed to provide the information required by the development staff involved only a small set of the candidate services discovered from the database. Indeed, only 73 services were published in total. The percentage of services that were published to support the functionalities of the CI portal was therefore 27.9%. Figure 14 provides a pyramid chart to show the amount of services published for each kind of service.

The service usage ratio depends on the project's goals (i.e., it depends on those business functionalities that must be supported in the modernized version of the information system). Despite this fact, this case study assumes that the required functionalities of the project under study are the same as those for which the legacy database was created (no additional functionalities are considered). This is due to the fact that the project motivation was the migration and integration of the legacy database into a SOA environment, because the previous data management consisted of the direct manual editing of tables through the database management system. Other modernization

Table VI. Planned and actual effort involved in the development of the CI portal's papers module.

Development activity	Planned effort (h)	Actual effort (h)	Effort gain
Create database schema	8	0	11.4%
Legacy data migration	30	0	42.9%
Application design	24	36	−17.1%
Application implementation	44	8	51.4%
Unitary tests	8	0	11.4%
Integration tests	8	8	0%
Total	122	52	—

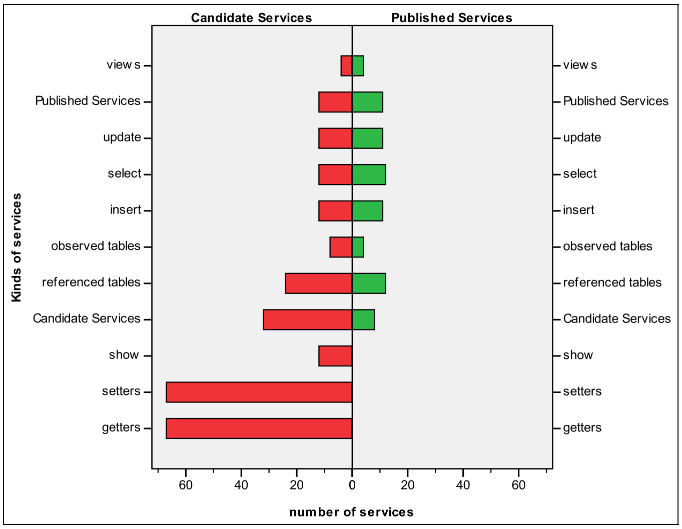


Figure 14. Performance of published services in CI portal.

projects might therefore need a different amount of Web services to support different functionalities. In fact, the development staff noticed that the nonselected Web services would be very useful for future developments. Because these nonconsidered services were identified and collected, it would be easy to deploy and integrate them into the CI Web application for the implementation of additional features.

The result is an operational Web service that handles the legacy database. The Web service supports the information needs in a SOA context such as a CI portal. At this point, the CI portal can carry out the required functionalities by means of the new Web services. As a consequence, the AQ1 question can be positively answered, that is, PRECISO can obtain Web services to manage all the stored data.

Furthermore, the AQ2 question must be also answered, and the effort time was thus evaluated. The total effort time previously planned was 122 h (work hours of a single developer). However, the actual effort time was only 52 h, signifying that the automatic generation of Web services provided a total effort reduction of 70 h (i.e., 57%).

The development staff perceived advantages in most of the development activities (see Figure 15). Because all the required information was available as services, the staff worked with real data when developing the Web application. The staff strove to develop the Web interface (with an effort gain of 51.4%). In addition, because the required information was available from the beginning, all the features of the CI Web application were tested with the real information from the database. The staff was allowed to reduce the time spent on the testing process (effort gain of 11.4%), because Web developers were able to build the necessary Web interfaces that display the information, which is in turn obtained by the aforementioned Web services.

The effort time was only worse in the application design activity (effort gain of −17.1%) because a small additional effort was necessary to plan the appropriate integration between Web services and the CI portal (Figure 15). This additional effort would not be necessary in traditional software developments that are aligned with the remainder of the CI portal's design. Despite this additional effort, which is small in comparison with the estimated time for the design activity, the total effort time was much better when the information requirements were implemented using PRECISO. The AQ2 question can therefore be answered as true, that is, PRECISO can efficiently obtain Web services. As a consequence, the main research question (MQ) is positively answered. This means that PRECISO is able to properly obtain Web services from legacy relational databases.

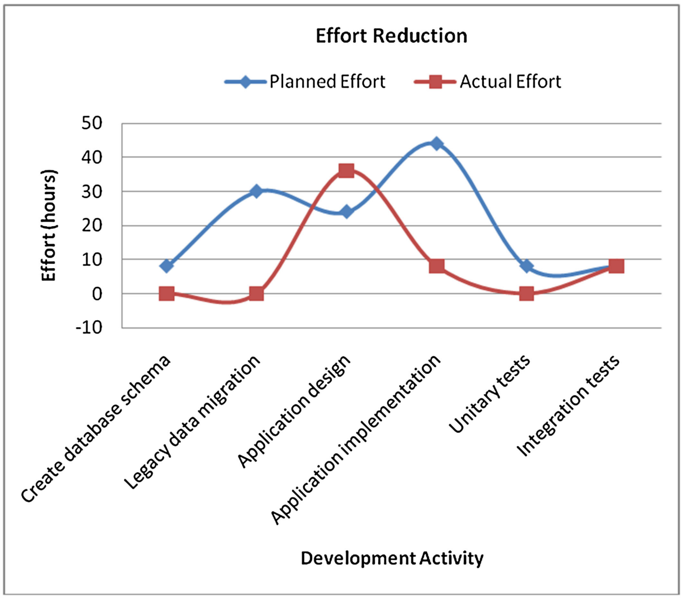


Figure 15. Planned and actual effort chart to develop the CI portal's module.

DOI: 10.1002/smr

6.7. Validity evaluation

Finally, the validity of the results obtained in the case study must be evaluated. This stage evaluates whether the results are true and not biased for the whole population to which the results will be generalized. This section thus shows the threats to the validity of this case study. According to Ref. [47], there are three main types of validity: internal, construct, and external validity.

There is no large population that makes it possible to obtain statistically representative results according to the internal validity, although a clear trend for the proposed measures was identifiable in this case study. However, there are two determining factors involved in the obtained results related to the efficiency of PRECISO. First, the tool used to obtain the business processes could be a factor that affects the effort time. This means that the measure values might be different if the Web services are obtained with another tool supporting the same proposed ADM-based process. Second, if this study is replicated with other cases involving a different software development plan, the results concerning effort time might be slightly different. However, a positive effort gain is expected in the majority of cases, because the proposed automation process allows developers to reduce the development time for several activities.

The measures of the case study were adequate to measure the variables and answer the research questions appropriately. Thus, the construct validity was also achieved. Finally, external validity is concerned with the generalization of the results. This study considers traditional legacy relational databases as the whole population. In this respect, the results obtained could be generalized to this population. However, the specific technology of the selected case (MSSQL Server 2000) is a threat that should be noted, because future results might be different. To mitigate this threat, the study should be replicated using legacy databases implemented in other different technologies like Oracle, MySQL, and so on.

7. CONCLUSION AND FUTURE WORK

This paper presented an ADM whose aim is to automatically generate Web services from legacy relational databases. Databases can thus be integrated into SOA environments. A tool (PRECISO) that supports this process has also been developed in an ad hoc manner.

The modernization process is based on the usage of a set of metamodels to represent the models involved in each task. For example, the SQL-92 metamodel has been used to represent the database model (or PSM model). Moreover, the UML2 metamodel has been utilized to represent the system (or PIM model), among other metamodels.

To illustrate how the process works, this paper also presented a case study in an industrial context. The tool developed was used to support the modernization process of a publication database, so that it could be used by different applications, among them a corporate website, which was built using the Web services generated. This case study revealed that developers took advantage of the availability of the required information to improve the development process: because all the required information was available as services, the team could work with real data when developing the Web application. In addition, the development team could work only on the development of the Web interface. Furthermore, because the required information was available from the beginning, all the features of the CATEDRA INDRA website could be tested with the real information from the database. This allowed the development team to accelerate the testing process.

The future lines of this research will focus on two key aspects: (1) an in-depth analysis will be carried out to infer services based on the searching of more patterns in a database scheme; (2) transformations among models will be formalized through the use of specific-purpose languages such as QVT; and (3) a top–down generation of Web services based on data from business processes will be added to obtain complex services in line with business requirements. In addition, new versions of the PRECISO tool supporting the new advances in this research will be implemented.

DOI: 10.1002/smr

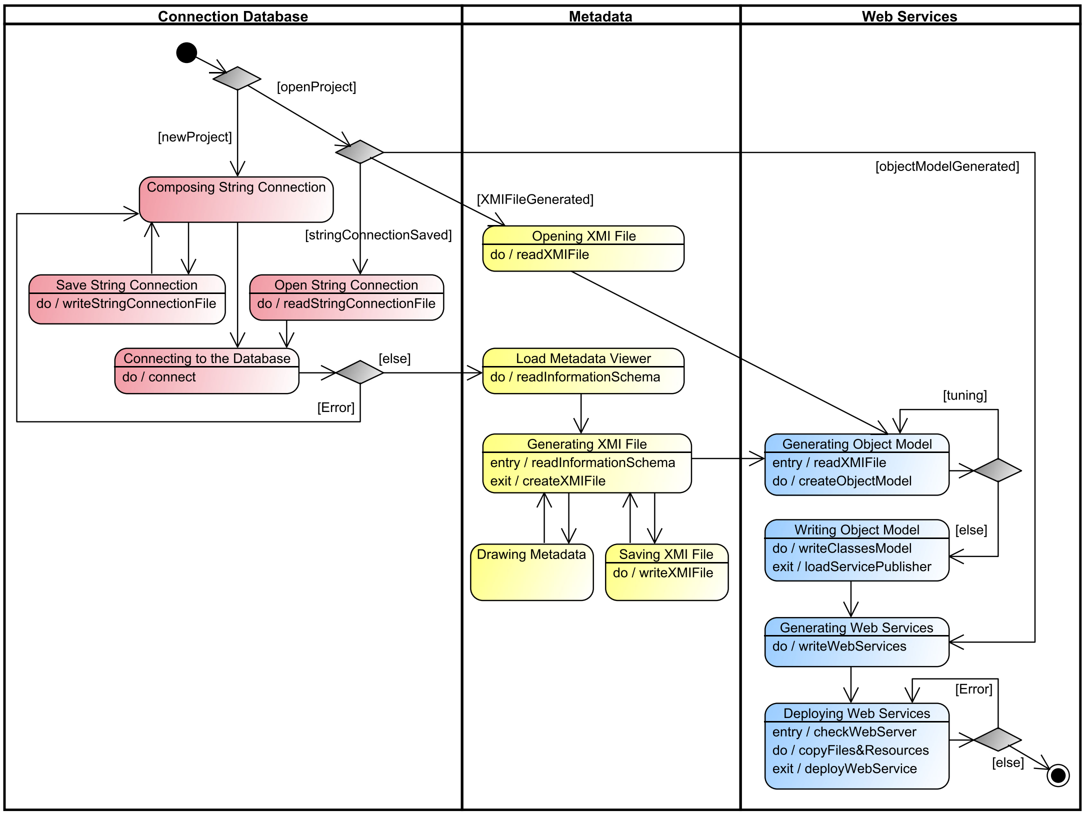
APPENDIX A: PRECISO OVERVIEW

Table A.I. Overview of PRECISO process. Activities, tasks, inputs, outputs, and techniques.

Activity	Task	Inputs	Outputs	Techniques	*Pre Tasks*	*Post Tasks*
Database Model Recovery	DMR1. Database reverse engineering	• Database schema	• Database model	• INFORMATION _SCHEMA recovery		• *DMR2* • *OMG1*
	DMR2. Service discovery	• Database schema	• Candidate services list	• Pattern matching	• *DMR1*	• *OMG1* • *WSG1*
Object Model Generation	OMG1. Object model transformation	• Database model	• Object model	• QVT transformations • ATL transformations • Programming transformations	• *DMR1* • *DMR2*	• *WSG1* • *WSG2*
Web services Generation	WSG1. Generation of WSDL interfaces	• Object model • Candidate services list	• WSDL model	• QVT transformations • ATL transformations • Programming transformations	• *DMR2* • *OMG1*	• *WSG3*
	WSG2. Generating code of object model	• Object model	• Source code	• Source code generation	• *OMG1*	• *WSG2*
	WSG3. Web services publication	• WSDL model • Source code	• Web services	• Visualization and selection • Source code generation	• *WSG1* • *WSG2*	• *WSG4*
	WSG4. Web services deployment	• Web services	• Web services deployed	• Deployment techniques • Web server configuration	• *WSG3*	

APPENDIX B: PARTITIONED PROCESS IN PRECISO

Figure B.I. Partitioned process in PRECISO tool.



DOI: 10.1002/smr

ACKNOWLEDGEMENTS

This work was supported by the Spanish FPU Program, and by the R + D projects ALTAMIRA (JCCM, PII2I09-0106-2463); PEGASO/MAGO (MICIN and FEDER, TIN2009-13718-C02-01); MOTERO (JCCM and FEDER, PEII11-0366-9449); ORIGIN (CDTI and FEDER, IDI-2010043(1-5)); and IQMNet (MICIN, TIN2010-09809-E).

REFERENCES

1. Shankaranarayanan G, Cai Y. *A Web Services Application for the Data Quality Management in the B2B Networked Environment*. In Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 7 - Volume 07. IEEE Computer Society, 2005.
2. Sommerville I. *Software Engineering* (8th edn). Addison Wesley: Reading, Massachusetts, 2006; 864.
3. Canfora G, Penta MD, Cerulo L. Achievements and challenges in software reverse engineering. *Communications of the ACM* 2011; **54**(4):142–151.
4. Davenport TH. Need radical innovation and continuous improvement? Integrate process reengineering and TQM. *Strategy & Leadership Journal* 1993; **21**(3):6–12.
5. Di Lucca GA, Fasolino AR, Tramontana P. Reverse engineering Web applications: the WARE approach. *Journal of Software Maintenance and Evolution: Research and Practice* 2004; **16**:71–110.
6. Sneed HM. Migrating to Web Services. In *Emerging Methods, Technologies and Process Management in Software Engineering*. Wiley-IEEE Computer Society Pr.: Hoboken, New Jersey, 2008; 151–176.
7. Lehman MM. On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software* 1979; **1**:213–221.
8. Chikofsky EJ, Cross JH. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software* 1990; **7**(1):13–17.
9. Miller J, Mukerji J. MDA Guide Version 1.0.1. [cited 05-12-2008]; Available from: www.omg.org/docs/omg/03-06-01.pdf 2003: OMG. 62.
10. OMG. ADM Glossary of Definitions and Terms. [cited 25-11-2009]; Available from: http://adm.omg.org/ADM_Glossary_ Spreadsheet_pdf.pdf. 2006, OMG. p. 34.
11. Caballero I, *et al*. IQM3: information quality management maturity model. *Journal of Universal Computer Science* 2008; **14**(22):3658–3685.
12. Lewis GA, Smith DB, Kontogiannis K. A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems. 2010, Software Engineering Institute. p. 40.
13. Arnold RS. *Software Reengineering*. IEEE Computer Society Press: Washington, D.C., 1994; 688.
14. Sneed HM. *Estimating the Costs of a Reengineering Project*. Proceedings of the 12th Working Conference on Reverse Engineering. IEEE Computer Society, 2005; 111–119.
15. Canfora G, Penta MD. New Frontiers of Reverse Engineering. In *2007 Future of Software Engineering*. IEEE Computer Society: Washington, D.C., 2007.
16. Kazman R, Woods SG, Carrière SJ. *Requirements for Integrating Software Architecture and Reengineering Models: CORUM II*. In Proceedings of the Working Conference on Reverse Engineering (WCRE'98). IEEE Computer Society, 1998.
17. OMG. Why do we need standards for the modernization of existing systems? 2003, OMG ADM Task Force.
18. OMG, QVT. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. [cited 10-10-2010]; Available from: http://www.omg.org/spec/QVT/1.0/PDF. 2008, OMG.
19. Turner M, Budgen D, Brereton P. *Turning Software into a Service*. IEEE Computer Society: Washington, D.C., 2003.
20. Reus T, Geers H, Deursen Av. Harvesting Software for MDA-Based Recovering. In *European Conference on Model Driven Architecture - Foundations and Applications*. Springer-Verlag Berlin Heidelberg: Bilbao (Spain), 2006.
21. Pereira ÓNM, Pinto JMHdS. *Maintainability assessment of an enhanced object-oriented approach for wrapping stored procedures*. In Proceedings of the 24th IASTED international conference on Database and applications. ACTA Press: Innsbruck, Austria, 2006; 26–31.
22. Behm A, Geppert A, Dittrich K. Algebraic Database Migration to Object Technology. In *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg: Berlin, Germany, 2000.
23. Hainaut J-L, *et al*. Database reverse engineering: From requirements to CARE tools. In *Applied Categorical Structures*. SpringerLink: Berlin, Germany, 2004.
24. Polo M, *et al*. Generating three-tier applications from relational databases: a formal and practical approach. *Information and Software Technology* 2002; **44**:923–941.
25. Thiran P, *et al*. *Updating Legacy Databases through Wrappers: Data Consistency Management*. In Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04) - Volume 00. IEEE Computer Society, 2004; 58–67
26. McBrien P, Poulovassilis A. *Automatic Migration and Wrapping of Database Applications - A Schema Transformation Approach*. In Proceedings of the 18th International Conference on Conceptual Modeling. Springer-Verlag, 1999.
27. Thiran P, Hainaut J-L. *Wrapper Development for Legacy Data Reuse*. In Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01). IEEE Computer Society, 2001; 198
28. Canfora G, *et al*. A wrapping approach for migrating legacy system interactive functionalities to Service Oriented Architectures. *Journal of Systems and Software* 2008; **81**(4):463–480.

DOI: 10.1002/smr

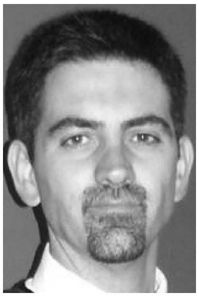
29. Chung S, Young PS, Nelson J. *Service-Oriented Software Reengineering: Bertie3 as Web Services*. In Proceedings of the IEEE International Conference on Web Services. IEEE Computer Society, 2005.
30. Bezivin J, *et al*. *Applying MDA Approach for Web Service Platform*. In Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International. IEEE Computer Society, 2004; 58–70.
31. Wang H, Shen B, Chen C. Model-Driven Reengineering of Database. In *World Congress on Software Engineering*, Beijun S, Cheng C (eds.), 2009; 113–117.
32. García-Rodríguez de Guzmán I, Polo M, Piattini M. *Using Model-Driven Pattern Matching to derive functionalities in Models*. In Proceedings of the Nineteenth International Conference on Software Engineering and Knowledge Engineering. Boston, USA, 2007; 529–534.
33. ISO/IEC. ISO/IEC 9075:1992, Database Language SQL. 1992.
34. OMG. Unified Modeling Language: Superstructure. Version 2.0. http://www.omg.org/docs/formal/05-07-04.pdf. 2007 [cited 16-08-2007]; Available from: http://www.omg.org/docs/formal/05-07-04.pdf.
35. Calero C. An Ontological Approach To Describe the SQL:2003 Object-Relational Features. *Computer Standards and Interfaces* 2005; 695–713.
36. Melton J, Simon AR. *Understanding the new SQL: A Complete Guide*. Morgan Kaufmann Publishers, Inc: United States of America, 1993.
37. Grose TJ, Doney GC, Brodsky SA. *Mastering XMI: Java Programming with XMI, XML, and UML*, Press O (ed). John Wiley & Sons: Hoboken, New Jersey, 2001; 480.
38. Larman C. *Applying UML and Patterns*. Prentice Hall: Upper Saddle River, New Jersey, USA, 1998.
39. W3C. WSDL in Web Services Description Working Group. 2007 [cited 08/01/2008]; Available from: http://www.w3.org/2002/ws/desc/.
40. Alarcos Research Group. PRECISO. A Reverse Engineering Tool to Discover Web Services from Relational Databases. 2009 [cited 5/30/2011]; Available from: http://alarcos.esi.uclm.es/per/rpdelcastillo/PRECISO/PRECISO.html.
41. Grose T, Doney G, Brodsky A. *Mastering XMI. Java Programming with XMI, XML, and UML*. John Wiley & Sons: Hoboken, New Jersey, 2002; 480.
42. Zimmermann O, Tomlinson M, Peuser S. Perspectives on Web Services. *Appling SOAP, WSDL and UDDI to Real-World Projects*. Springer: Berlin, Germany, 2003.
43. Blaha M. *A Retrospective on Industrial Database Reverse Engineering Projects-Part 1*. In Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01). IEEE Computer Society: Suttgart, Germany, 2001.
44. Blaha M. *A Retrospective on Industrial Database Reverse Engineering Projects-Part 2*. In Proceedings of the 8th Working Conference on Reverse Engineering (WCRE´01). IEEE Computer Society: Suttgart, Germany, 2001.
45. Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Eng.* 2009; **14**(2):131–164.
46. Yin RK. *Case study research. Design and methods* (3rd edn). Sage: London, 2003.
47. Wohlin C, *et al*. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers: Norwell, Massachusetts, 2000; 204.

AUTHORS' BIOGRAPHIES

Ricardo Pérez-Castillo holds the MSc degree in Computer Science from the University of Castilla-La Mancha, and he is currently a PhD student in Computer Science. He Works at the Instituto de Tecnologías y Sistemas de Información (ITSI) at the University of Castilla-La Mancha. His research intererests include architecture-driven modernization, model-driven development and business process mining. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ricardo.PdelCastillo@uclm.es.

Ignacio García-Rodriguez de Guzmán is assistant professor at the University of Castilla-La Mancha and belongs to the Alarcos Research Group at the UCLM. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software maintenance, software modernization and service-oriented architecture. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ignacio.GRodriguez@uclm.es.

DOI: 10.1002/smr



Ismael Caballero is assistant professor at the University of Castilla-La Mancha and belongs to the Alarcos Research Group at the UCLM. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software and data quality, database design and software development based on quality. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Ismael.Caballero@uclm.es.

Mario Piattini is full professor at the UCLM. His research interests include software quality, metrics and maintenance. He holds the PhD degree in Computer Science from the Technical University of Madrid, and leads the Alarcos Research Group at the Universidad de Castilla-La Mancha. He is CISA, CISM e CGEIT by ISACA. Contact him at Escuela Superior de Informática, Paseo de la Universidad 4, 13071-Ciudad Real, Spain; Mario.Piattini@uclm.es.

DOI: 10.1002/smr