Science of Computer Programming 97 (2015) 113–121

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Understanding database schema evolution: A case study

Anthony Cleve [a,*], Maxime Gobert [a], Loup Meurice [a], Jerome Maes [a], Jens Weber [b]

[a] University of Namur, Belgium
[b] University of Victoria, Canada

HIGHLIGHTS

- We present a tool-supported method to analyze the history of a database schema.
- The method makes use of mining software repositories (MSR) techniques.
- We report on the application of the method to a large-scale case study.

ARTICLE INFO

Article history:
Received 9 October 2013
Accepted 5 November 2013
Available online 22 November 2013

Keywords:
Database understanding
Schema evolution
Software repository mining

ABSTRACT

Database reverse engineering (DRE) has traditionally been carried out by considering three main information sources: (1) the database schema, (2) the stored data, and (3) the application programs. Not all of these information sources are always available, or of sufficient quality to inform the DRE process. For example, getting access to real-world data is often extremely problematic for information systems that maintain private data. In recent years, the analysis of the evolution history of software programs have gained an increasing role in reverse engineering in general, but comparatively little such research has been carried out in the context of database reverse engineering. The goal of this paper is to contribute to narrowing this gap and exploring the use of the database evolution history as an additional information source to aid database schema reverse engineering. We present a tool-supported method for analyzing the evolution history of legacy databases, and we report on a large-scale case study of reverse engineering a complex information system and curate it as a benchmark for future research efforts within the community.

Working in Paul Klint's group at CWI has been my very first professional experience. This was certainly the best possible way to start my research career. Paul is one of the most inspiring persons I have ever met. His ability to share his enthusiasm with his colleagues and students, and to show them the way through his own achievements, is truly outstanding. Now that I have, in turn, the privilege to supervise students, I make sure to regularly ask them the question Paul used to ask me: "So, are you still making progress?" Their most recent answer is summarized in this paper. Happy Birthday, Paul! – Anthony.

1. Introduction

Understanding the evolution history of a complex software system can significantly aid and inform current and future development initiatives of that system. Software repositories such as version management systems and issue trackers provide excellent opportunities for historical analyses of system evolution. Most research work in this area has concentrated on program code, design and architecture. Fewer studies have focused on database systems and schemas. This is an unfortunate gap as databases are often at the heart of many of today's information systems. Understanding the database schema –

* Corresponding author.

0167-6423/$ – see front matter
http://dx.doi.org/10.1016/j.scico.2013.11.025

which captures domain-specific concepts, data structures and integrity constraints – often constitutes a prerequisite to understanding the evolution of such systems.

In this paper, we report on our experiences made in the context of a real-world project with the objective of evolving a complex medical information system to fit new requirements. Specifically, (1) we present the tool-supported approach we developed to better understand the evolution history of the system's database, (2) we identify research challenges in the context of studying the evolution of data-intensive systems, and (3) we curate a rich and complex case study that can be used to explore these challenges (and others) by the software evolution research community.

The remainder of this paper is structured as follows. The next section introduces the main subject system studied in this paper (OSCAR) and the general context of our software evolution project. Section 3 describes the approach we have followed to study the evolution of the OSCAR database. In Section 4, we briefly present the tool suite that supports our approach. The results obtained when analyzing the OSCAR's history are summarized in Section 5 and discussed in Section 6. A related work discussion is given in Section 7 and Section 8 provides concluding remarks.

2. Context: The OSCAR system

OSCAR (*Open Source Clinical Application Resource*) is full-featured Electronic Medical Record (EMR) software system for primary care clinics. It has been under development since 2001 and is widely used in hundreds of clinics across Canada. As an open source project, OSCAR has a broad and active community of users and developers. The Department of Family Practice at McMaster University, which has managed OSCAR development efforts from inception to 2012, has recently transferred oversight of ongoing development to a newly formed not-for-profit company called OSCAR-EMR. This move was motivated by a new regulatory requirement to undergo ISO certification (*ISO 13485 Medical devices – Quality management systems*).

OSCAR architecture. OSCAR has a Web application architecture following the classical 3-tier paradigm. It employs a Java-based technology stack, making use of Java Server Pages (JSP), Enterprise Java Beans (J2EE) and several frameworks such as Spring, Struts and Hibernate. The source code comprises approximately two million lines of code with a rough distribution of 600 kLOC for the application logic, 1200 kLOC for the presentation layer and 100 kLOC for the persistence layer. OSCAR uses MySQL as the relational database engine and a combination of different ways to access it, including Hibernate object-relational middleware, Java Persistence Architecture (JPA) and dynamic SQL (via JDBC). The reason for this combination of technologies is the constant and ongoing evolution history of the product, which originated from JDBC, via Hibernate to JPA.

Oscar database. The OSCAR database schema has over 440 tables and many thousands of attributes. At the time of conducting our study, the database schema of the OSCAR distribution did not contain any information on relationships between tables (foreign keys) and no documentation was available about the schema. We later learned that the missing relationships were due to the evolution history of OSCAR, which has been using the older MyISAM database engine provided by MySQL that does not support foreign keys. A port to the newer InnoDB engine is underway, which will eventually allow foreign keys to be defined explicitly.

OSCAR software repositories. The OSCAR community utilizes a range of software repositories and tools, including a feature request and bug tracking system (provided by Sourceforge), a source code submission and review system (Gerrit Code Review), a git-based configuration management system, a community Wiki (based on Plone) and three active mailing lists (one for developers and two for users of different levels of technical expertise).

The need to understand the database schema. The OSCAR database has grown organically over many years and knowledge about its internal structure is distributed among pockets of developers who have been contributing to specific functions of the system (e.g., prescription writer, representation of lab results etc.). Our need to understand the OSCAR database schema originated from our involvement in a project with the goal to develop software for a primary care research network (PCRN). The purpose of the PCRN is to integrate health information kept in primary care EMR software in order to make them accessible to medical research and data mining. An important step in developing the PCRN software is to create "*export conduits*" for transferring health data from the EMR into a research database for subsequent query processing. Due to its popularity (second largest market share in British Columbia) and openness, OSCAR has been chosen as one of the first EMR products to interface with the emerging PCRN.

While designing early versions of the PCRN data migration adapter for OSCAR, we found that we were running into questions pertaining to the database schema. Of course, as could be expected for any heavily evolved, real-world system, some of them had to do with the fact that the database schema lacked documentation. Moreover, the schema did not contain any declared relationships (foreign keys). Other questions were of a more semantic and puzzling nature. For example, when attempting to design the function to export data on patient immunization records, we found two seemingly unrelated schema structures covering the same semantic issue. One schema structure revolved around tables entitled "*immunizations*" and "*configimmunization*" while the other schema structure revolved around tables entitled "*preventions*" and "*preventionsext*". During our project we found that taking into consideration the evolution history of the database schema was helpful in answering questions like these. (We found out that the "prevention" structured superseded the "immunization" structure but has still been retained in order to deal with legacy data.) This motivated us to investigate more formally OSCAR's evolution history and develop methods and tools to help with this investigation.