Co-evolving code-related and database-related activities in a data-intensive software system

Mathieu Goeminne, Alexandre Decan, Tom Mens
Département Informatique, Université de Mons
7000 Mons, Belgique
Email: firstname.lastname@umons.ac.be

***Abstract*—Current empirical studies on the evolution of software systems are primarily analyzing source code. Sometimes, social aspects such as the activity of contributors are considered as well. Very few studies, however, focus on data-intensive software systems (DISS), in which a significant part of the total development effort is devoted to maintaining and evolving the database schema. We report on early results obtained in the empirical analysis of the co-evolution between code-related and database-related activities in a large open source DISS. As a case study, we have analyzed OSCAR, for which the historical information spanning many years is available in a Git repository.**

I. Introduction

In [1] we presented some important challenges developers face during the evolution of *data-intensive software systems* (DISS). These systems are composed of a database, typically managed by a database management system, and the source code that implements the main functionality of the system. Adding new functionality to the system may affect the database structure and, conversely, modifying the database structure may impact the source code associated to it. Because of this, evolution of the DISS requires the source code and the database schema to co-evolve.

If, in addition, the DISS is an open source system that is developed in a distributed way by a large team of developers over a long period of time, the need for understanding, supporting and improving this co-evolution process becomes even more important. This requires us to find answers to a whole range of questions. Is it better to have a development team in which everyone modifies both the source code and database structure, or should one separate concerns, with a group of developers working primarily on the source code functionality, and another group being primarily in charge of the database changes? What is the impact of introducing a new database technology? How do developers divide their effort between the activity types involved in evolving a DISS?

Very little in-depth longitudinal empirical studies have been carried out to study the co-evolution of source code and database schemas of DISS. In this paper we aim to provide such an analysis on the basis of OSCAR, a non-trivial DISS.

II. Related Work

There is quite some empirical research on co-evolution between different types of artefacts constituting a software system. For example, Zaidman et al. [2] have empirically studied the co-evolution of production code and test code, in both an open source and an industrial software system. Fluri et al. [3] empirically studied the co-evolution between source code and associated comments.

Concerning the co-evolution between database schemas and source code in DISS, much less research is available. The most relevant one that we are aware of is the comprehensive empirical analysis by Qiu et al. [4] on the co-evolution of code and database schemas in ten open-source database applications.

III. About OSCAR

SCOOP[1] (Social Collaboratory for Outcome Oriented Primary care) is a Canadian research network aimed at providing a tool infrastructure to answer questions related to Electronic Medical Records (EMR) used in primary care. To this aim, SCOOP is in charge of developing several open source projects, available on Github[2]. One of these tools is OSCAR[3], an open source EMR system designed to improve healthcare. It is used by many private physicians to capture individual information through a dedicated user interface, and currently supports over 1.5 million patients across Canada. The OSCAR ecosystem is managed by OSCAR EMR[4], a not-for-profit corporation representing the OSCAR community consisting of research institutions, healthcare institutes, service providers and patients. OSCAR EMR is in charge of ensuring quality, implementation and support of the OSCAR product suite.

The OSCAR product suite contains a range of applications and interfaces. *OSCAR* itself is an EMR system containing healthcare information with full billing capabilities, chronic disease management tools, prescription module, scheduling and much more. *OSCAR CAISI* is a case management, bed management and program facility management system to share relevant patient information among providers, while maintaining a high level of patient privacy. It is used for helping vulnerable patients such as homeless people.

The main characteristics of OSCAR, mined from its Git repository, are summarised in Table I. The process used for

[1] http://scoop.leadlab.ca/
[2] github.com/scoophealth
[3] github.com/scoophealth/oscar
[4] www.oscar-emr.com/

obtaining this and more detailed information is explained in Section IV.

characteristic	value
duration	3,939 days (>129 months)
start date	8 November 2002
end date	21 August 2013
number of commits	18,727
number of file touches	93,721
number of *distinct* contributors	100
number of distinct files	20,718
number of distinct Java files	7,273
number of distinct JSP files	3,918

TABLE I
SOME BASIC METRICS FOR OSCAR, MEASURED OVER THE ENTIRE CONSIDERED PERIOD.

OSCAR has been implemented primarily in Java and JSP (Java Server Pages[5]). Fig. 1 shows the monthly aggregated proportion of Java and JSP files over time (all other file types are excluded). Initially, the proportion of JSP files is very high (70%) but this proportion gradually decreases until it reaches 29% in the last studied revision.

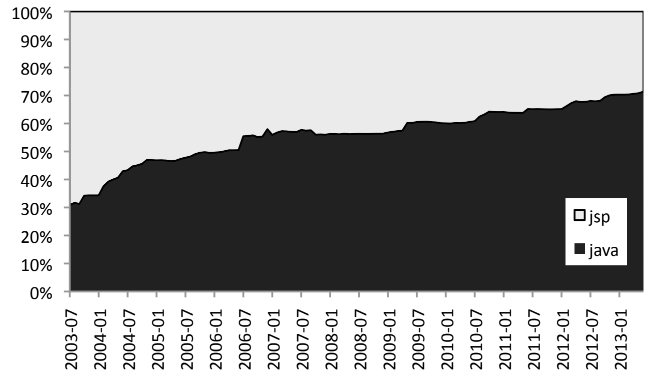

Fig. 1. Evolution of proportion of Java and JSP files in OSCAR.

IV. EXPERIMENTAL SETUP

A. Research Questions

Our main goal is to empirically study the co-evolution between source code changes and database schema changes in a data-intensive software system. Using OSCAR as our case study, we aim to explore the following research questions:

RQ1 Is there any co-evolution between source code changes and database-related changes?
RQ2 What is the impact on co-evolution of introducing a particular database technology?
RQ3 How do contributors divide their work and how does this evolve over time? (E.g. Are there distinct teams of contributors involved in the source code and the database?)

[5]jsp.java.net

B. Data extraction

We first extracted all commits in the OSCAR Git repository mirror[6] using two complementary tools. We used CVS-Analy2[7] to extract all relevant information about file changes for OSCAR and to store it in a FLOSSMetrics-compliant database [5]. Our research collaborators, members of the PReCISE group at the University of Namur, used DB-Main[8] to deduce the database schema being in effect at any point of the project history, on the basis of the following SQL files contained in the `database/mysql/` folder of the source code repository: `oscarinit_on.sql`, `oscarinit.sql`, `icd10.sql` and `initcaisi.sql`. We merged the data gathered by both of these extraction tools into a single SQL database that we considered as our representation of the full project history of OSCAR. This database was iteratively enriched with extra views and tables to facilitate answering our research questions.

C. Identity merging

Over the entire considered period, 100 distinct contributors have been involved in OSCAR. Fig. 2 shows the number of distinct contributors involved on a monthly basis.

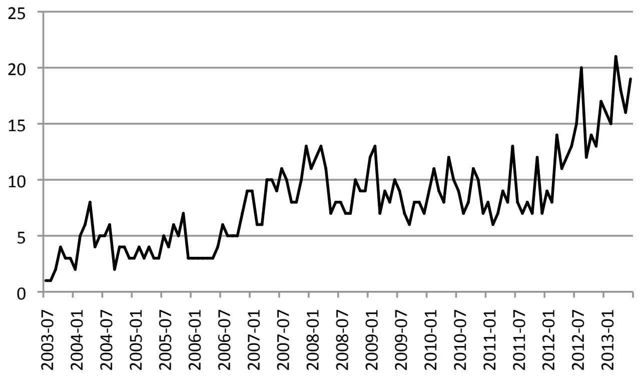

Fig. 2. Monthly number of distinct active contributors to OSCAR.

The identification of distinct contributors was based on the Git *author* accounts consisting of a login and an optional e-mail address) that are required for each commit. Since the same contributor may use a different account for different commits, we merged the accounts associated to the same contributor into a single *identity* in order to reflect the actual activity of each contributor [6]. This identity merging was achieved by comparing, for each contributor account, the logins and prefixes of e-mail addresses, which we refer to as *labels* hereafter. The pairs of labels are ordered by increasing Levenshtein distance [7]. Starting from the top of the list, the contributor accounts are then merged together into the same identity, until a human supervisor decided that the Levenshtein distance became too high (i.e. the pairs of labels no longer

[6]https://github.com/scoophealth/oscar.git
[7]http://metricsgrimoire.github.io/CVSAnalY/
[8]http://www.db-main.eu/?q=en

correspond to the same contributor) and that a new identity should be created. Throughout the process, *false positives*, , *i.e.* pairs of labels that corresponded to a distinct contributor even though the Levenshtein distance was low, were filtered out manually.

V. Preliminary Results

In order to answer the research questions, we statistically analysed the information stored in our database using R.

A. Research Question 1

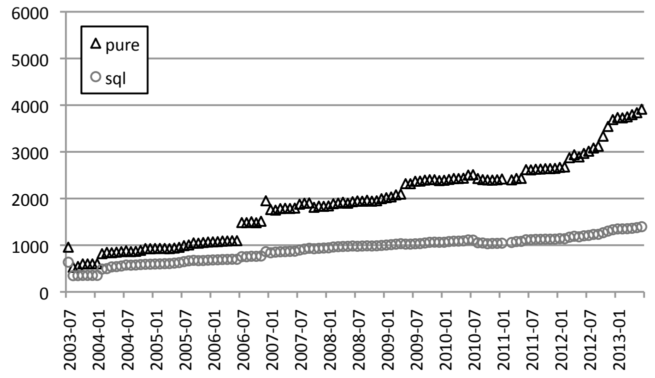

Fig. 3. Evolution of (monthly aggregated) number of database-unrelated files (indicated as *pure*) and files containing embedded SQL queries (indicated as *sql*).

For answering research question RQ1, we first visualised the growth of OSCAR over time (Fig. 3), along two dimensions: the monthly number of active *pure* (i.e., database-unrelated) source code files and the number of code files containing embedded SQL queries (*sql* in Fig. 3). A linear growth was confirmed in both cases, through a linear regression model with very good fit and statistical significance:
For *pure* code files: adjusted $R^2 = 0.948$, $p < 2.2e - 16$;
For *sql* code files: adjusted $R^2 = 0.948$, $p < 2.2e - 16$.

Next we computed Spearman's (ρ) and Kendall's (τ) rank correlation between both types of files. We observed a strong and statistically significant ($p < 2.2e - 16$) correlation: $\rho = 0.997$; $\tau = 0.970$.

B. Research Question 2

With RQ2 we want to study the impact of a particular database technology. Fig. 4 shows the monthly proportion of *pure* source code files (i.e., Java and JSP files that are not directly linked to the database) as opposed to *database-related* source code files (*sql*, *jpa*, *hxml*) that are related to the use of a particular database technology. While the proportion of pure files stays relatively stable over time, and always exceeds 60%, we observe a different pattern for the evolution database-related files.

In the beginning of the project, only raw SQL queries embedded in the source code (*sql* in Fig. 4) were used to get, add, update and remove information in the database. Starting from July 2006, the object-relational mapping framework

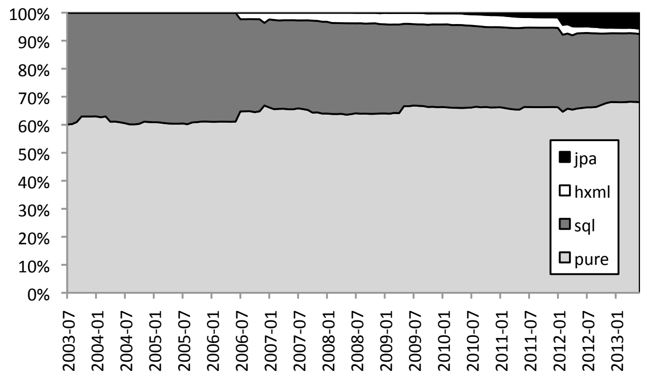

Fig. 4. Evolution of the relative proportion of file types in OSCAR.

Hibernate[9] was introduced. XML files were used to map Java classes to the database tables. These XML files describe the Java classes (*hxml* in Fig. 4) corresponding to a particular database table as well as the class properties corresponding to a particular database column. The types and other properties (such as the presence of an index or the relation between tables) may also optionally be described in these XML files. Starting from July 2008, JPA technology (*jpa* in Fig. 4) was introduced for Hibernate as an alternative to these XML files. JPA takes care of the mapping between classes (respectively class attributes) and database tables (respectively database columns) by adding particular annotations directly in the Java classes that represent the database tables. We observe in Fig. 4 that these JPA annotations start to replace the XML files gradually.

C. Research Question 3

To answer RQ3, we linked the type of activity (green = database-unrelated; red = database-related; blue = both) to individual OSCAR contributors. Fig. 5 visualises the result. We observe that in the large majority of cases, the monthly activity of individual contributors is a combination of database-unrelated and database-related changes. This indicates that there is no clear separation of concerns between pure coding activities and database-related activities.

We analysed this in more detail by taking the programming and database technology into account. We correlated the monthly numbers of active files (Table II) or file touches (Table III) for specific file types: *java* and *jsp* source-code files, files containing raw *sql* statements or *jpa* tags and source code files referred to by XML Hibernate mapping files (*hxml*). We also correlate these with the monthly number of active contributors. We observe a strong and statistically significant correlation for each considered combination, except for the active source code files referred by XML Hibernate mapping files (*hxml*). In the case of file touches, the results are similar, but the correlation is a bit weaker, especially between each file type and the number of contributors.

[9]hibernate.org

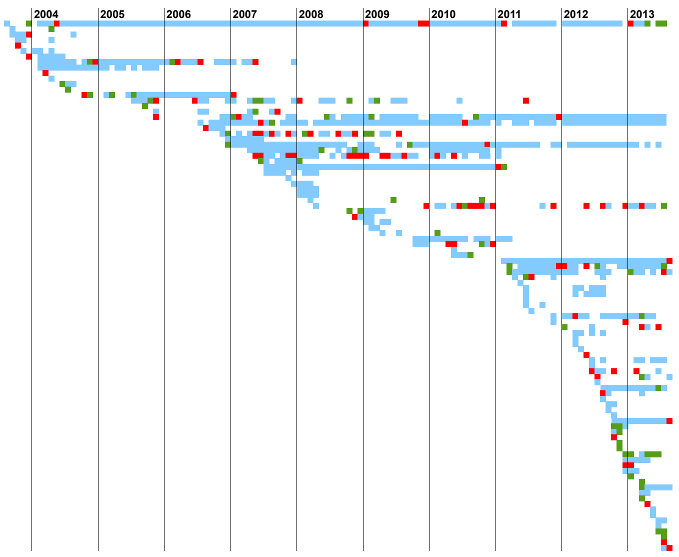

Fig. 5. Evolution of the (monthly) activity type per contributor involved in OSCAR: green indicates only database-unrelated changes; red indicates only database-related changes; blue indicates a combination of both types of changes.

	sql	jpa	hxml	java	jsp	pure
contributors	**0.79**	0.59	<u>-0.11</u>	**0.79**	**0.79**	**0.79**
pure	**1.00**	**0.97**	0.43	**1.00**	**0.96**	
jsp	**0.97**	**0.60**	0.36	**0.94**		
java	**0.99**	**0.99**	0.45			
hxml	0.42	<u>-0.30</u>				
jpa	**0.95**					

TABLE II
SPEARMAN RANK CORRELATIONS FOR ACTIVE FILES AND CONTRIBUTORS. STRONG CORRELATIONS ≥ 0.60 ARE IN **BOLDFACE**. VALUES <u>WITHOUT</u> STATISTICAL SIGNIFICANCE ($p > 0.01$) ARE <u>UNDERLINED</u>.

	sql	jpa	hxml	java	jsp	pure
contributors	0.51	0.45	0.13	**0.60**	0.49	0.58
pure	**0.88**	**0.60**	0.58	**0.95**	**0.85**	
jsp	**0.94**	**0.63**	0.50	**0.80**		
java	**0.87**	**0.65**	0.59			
hxml	0.48	<u>0.13</u>				
jpa	**0.67**					

TABLE III
SPEARMAN RANK CORRELATIONS FOR FILE TOUCHES AND ACTIVE CONTRIBUTORS. SAME LEGEND AS TABLE II.

We conclude that, with the exception of Hibernate mapping files, even at this lower level of granularity there is not a clear separation of concerns between database-related and database-unrelated activities.

VI. Threats to Validity and Future Work

Our analysis may be biased by limitations in the Git extraction [8], CVSAnaly2, DB-Main and measurement tools that were used. The identity merging we performed is not necessarily perfect, but we are quite confident that the merging has been done correctly in the large majority of cases.

The statistical observations we have made may be imprecise because our analysis was too coarse-grained. We therefore intend to study the particular types of changes (both in the source code and the database schema) to get a deeper insight in the co-evolution aspects.

The analysis and results reported for the OSCAR case study are probably not generalisable. We have not found another data-intensive open source system of comparable size and duration, so it remains an open question whether the patterns of co-evolution identified here are also apparent in other similar systems. If not, it would be worthwhile to study the rationale behind the differences found.

An interesting topic of future work would be to look at families of software systems that all work with the same database. If this is the case, modifications in the database schema may impact many systems.

VII. Conclusions

Our goal was to study the co-evolution between code-related and database-related activities in a data-intensive software system (DISS). To achieve this, we performed a case study on OSCAR, a large and long-lived DISS, that has been under active development for over 12 years by 100 different contributors.

Our results revealed that there is a strong co-evolution between the source code files and the database-related files, and that there is no strict separation of concerns: all contributors are making changes to both types of files. We observed some changes in the database technology used over time, but were not able to see an impact of this on the co-evolution.

Acknowledgment

This research was carried out in the context of FRFC PDR research project T.0022.13 ”Data-Intensive Software System Evolution”, in collaboration with the University of Namur, and financed by the F.R.S.-FNRS Fonds de la Recherche Scientifique.

References

[1] A. Cleve, T. Mens, and J.-L. Hainaut, “Data-intensive system evolution,” *IEEE Computer*, vol. 43, no. 8, pp. 110–112, 2010.

[2] A. Zaidman, B. Rompaey, A. van Deursen, and S. Demeyer, “Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining,” *J. Empirical Software Engineering*, vol. 16, no. 3, pp. 325–364, 2011.

[3] B. Fluri, M. Würsch, E. Giger, and H. C. Gall, “Analyzing the co-evolution of comments and source code,” *Software Quality Control*, vol. 17, no. 4, pp. 367–394, Dec. 2009.

[4] D. Qiu, B. Li, and Z. Su, “An empirical analysis of the co-evolution of schema and code in database applications,” in *Joint European Soft. Eng. Conf. and ACM SIGSOFT Int. Symp. on Foundations of Software Engineering*. ACM , 2013.

[5] I. Herraiz, D. Izquierdo-Cortazar, and F. Rivas-Hernández, “Flossmetrics: Free/libre/open source software metrics,” in *European Conf. on Software Maintenance and Reengineering*. IEEE Computer Society, 2009, pp. 281–284.

[6] M. Goeminne and T. Mens, “A comparison of identity merge algorithms for software repositories,” *Science of Computer Programming*, 2011.

[7] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[8] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. German, and P. Devanbu, “The promises and perils of mining Git,” in *Int. Working Conf. on Mining Software Repositories*, 2009, pp. 1–10.