Co-evolving Code-Related and Database-Related Changes in a Data-Intensive Software System

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

***Abstract*—Current empirical studies on the evolution of software systems are primarily analysing 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 analysed 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 functionalities 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. Does everyone in the development team modify both the source code and database structure, or is there a separation of responsibilities, 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 effect 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. Gall et al. [4] identified logical coupling, i.e., coupling that might not be immediately noticeable from analysing the source code, but coupling that is visible through the fact that certain entities are frequently committed together.

Concerning the co-evolution between database schemas and source code in DISS, much less research is available. Maule et al. [5] carried out a case study of a commercial object-oriented content management system to analyse the impact on object-oriented source code of changes made to the database schema. They used static program analysis for this purpose. Qiu et al. [6] carried out an empirical analysis on the co-evolution of database schemas and code in ten open-source database applications from various domains. They studied specific change types inside the database schema and the impact of such changes on the code. In contrast, we study the database co-evolution from a code-centric viewpoint, by exploring the evolution of database-related code files. In addition, we explore the social dimension by analysing the intersection of the developers involved in code-specific and database-specific changes.

III. About OSCAR

SCOOP (Social Collaboratory for Outcome Oriented Primary care, http://scoop.leadlab.ca) 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 (http://github.com/scoophealth). One of these tools is OSCAR, 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 a not-for-profit corporation (OSCAR EMR, http://www.oscar-emr.com) representing the community consisting of research institutions, healthcare institutes, service providers and patients. OSCAR EMR is in charge of ensuring quality, implementation and support of the product suite.

The OSCAR product suite contains a range of applications and interfaces. *OSCAR* itself is an EMR system containing

978-1-4799-3752-3/14 © 2014 IEEE

CSMR-WCRE 2014, Antwerp, Belgium
Early Research Achievement

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.

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

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* developers	100
number of distinct files	20,718
number of distinct Java files	7,273
number of distinct JSP files	3,918

The main characteristics of OSCAR, mined from its Git repository, are summarised in Table I. The process used for obtaining this and more detailed information is explained in Section IV. OSCAR has been implemented primarily in Java and JSP (Java Server Pages, http://jsp.java.net). 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.

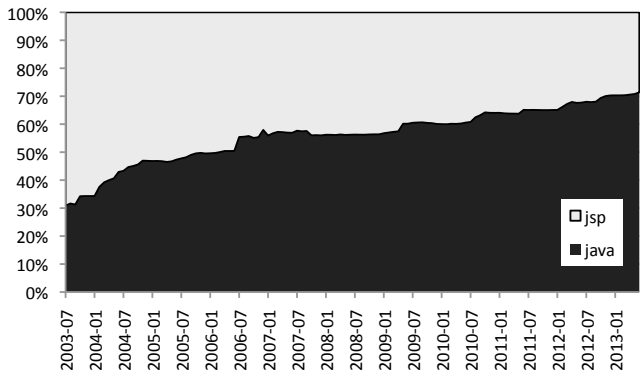

Fig. 1. Evolution of proportion of Java and JSP files in OSCAR.

IV. EXPERIMENTAL SETUP

A. Research Questions

Using OSCAR as our case study, it is our goal to empirically study the co-evolution between source code and database changes in a data-intensive software system. In software development, for maintainability reasons it is considered good practice to have a separation of responsibilities between different types of activities (in our case, code-related and database-related file changes). If everyone is involved in every activity, it introduces a communication overhead, and it is likely to be more difficult to migrate to a new database technology. For these reasons we explore the following research questions:
RQ_1: Is there any relation between how source code files and database-related files evolve?
RQ_2: What is the effect on co-evolution of introducing ORM and persistency technology?
RQ_3: How do developers divide their work and how does this evolve over time?
RQ_4: Who is involved in introducing (changes in) database-related code?

B. Data Extraction and Processing

Using a combination of Java and Python programs, we extracted all commits in the OSCAR Git repository mirror https://github.com/scoophealth/oscar.git. CVSAnaly2[1] was used to extract all relevant information about file changes and to store it in a FLOSSMetrics-compliant database [7]. This database was iteratively enriched with extra views and tables to facilitate answering our research questions.

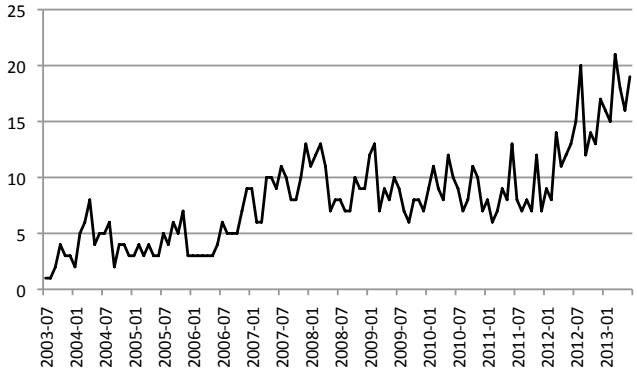

Fig. 2. Monthly number of distinct active contributors to OSCAR.

Over the entire considered period, 100 distinct developers have been involved in OSCAR. Fig. 2 shows the number of distinct developers involved on a monthly basis. The identification of these distinct developers 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 developer may use a different account for different commits, we merged the accounts associated to the same developer into a single *identity* in order to reflect the actual activity of each developer [8]. This identity merging was achieved by comparing, for each 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 [9]. Starting from the top of the list, the 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 correspond to the same contributor) and that a new identity should be created. Throughout the

[1]http://metricsgrimoire.github.io/CVSAnalY/

process, *false positives*, *i.e.* pairs of labels that corresponded to a distinct developer even though the Levenshtein distance was low, were filtered out manually.

V. Results

In order to answer the research questions, we quantitatively and visually analysed the information stored in our database using a combination of R and Python scripts.

Research Question 1

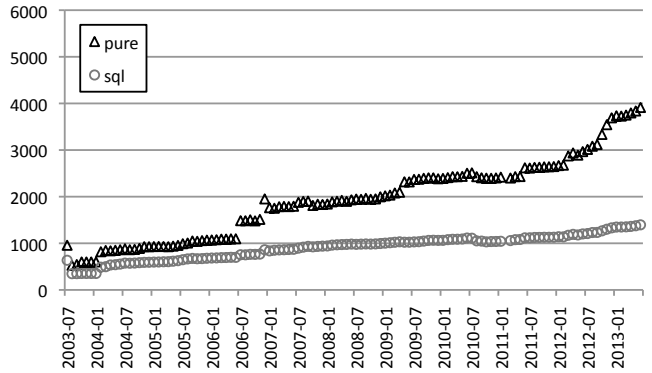

Fig. 3. Evolution of (monthly aggregated) number of database-unrelated files (indicated as *pure*) and files containing embedded SQL queries (indicated as *sql*).

For answering research question RQ_1, 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, the adjusted $R^2 = 0.948$ ($p < 0.01$) and for *sql* code files, the adjusted $R^2 = 0.948$ ($p < 0.01$). We also computed Spearman's (ρ) and Kendall's (τ) rank correlation between both types of files and observed a strong and statistically significant ($p < 0.01$) correlation: $\rho = 0.997$; $\tau = 0.970$.

Research Question 2

With RQ_2 we wish to study the effect of introducing a new database technology. Currently, we only do this at the granularity of files. In the beginning of the project, OSCAR only contained raw SQL queries embedded in the source code (*SQL* in Fig. 4) to get, add, update and remove information in the database. Starting from July 2006, the object-relational mapping (ORM) framework Hibernate (http://hibernate.org) was introduced. XML files were used to map Java classes to the database tables. These XML files describe the Java classes 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, the Java Persistence API (JPA) 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.

In the following, we denote by *SQL* a Java or JSP file containing at least one hard-coded SQL query, we denote by *HIB* a Java file that is the target of a mapping defined in some Hibernate XML configuration file, and we denote by *JPA* a Java file that contains at least one JPA annotation.

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 of type *SQL*, *HIB* or *JPA*. Note that files belonging to both the SQL and the HIB or JPA category, are only counted as belonging to the latter category, to reflect the fact that newer technologies dominate older ones. This implies that the number of files classified as *SQL* in Fig. 4 is a subset of the number of files classified as *sql* in Fig. 3.

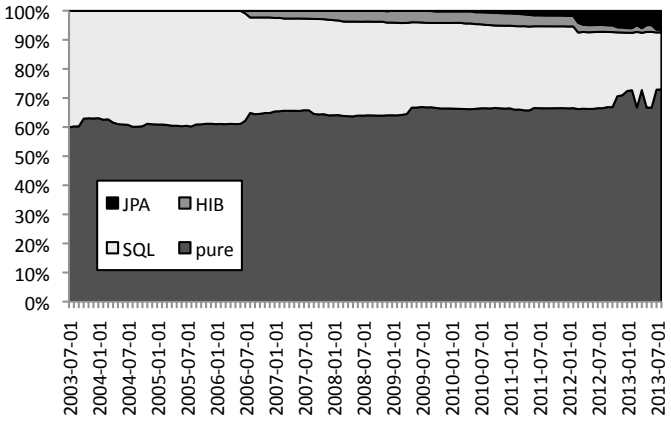

Fig. 4. Monthly evolution of the relative proportion of files of a given type in OSCAR.

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. Since July 2008, *JPA* files start to overtake and replace the *HIB* files gradually.

Fig. 5 displays the evolution in terms of absolute (as opposed to relative) number of database-related files only. Despite the introduction of Hibernate and JPA persistence providers, the number of code files containing embedded SQL stills remains quite high, indicating that the new technology has not fully replaced the old one. More precisely, even though the proportion of files containing embedded *SQL* gets smaller, it continues to stay around 80% of all database-related files.

We analysed this in more detail by looking at the number of *file touches* per activity type. We computed the Spearman rank correlation of the monthly aggregated numbers of file touches (Table II). We observe that touches to database-related files are strongly correlated to changes to database-unrelated files. SQL-type file touches are also strongly correlated to

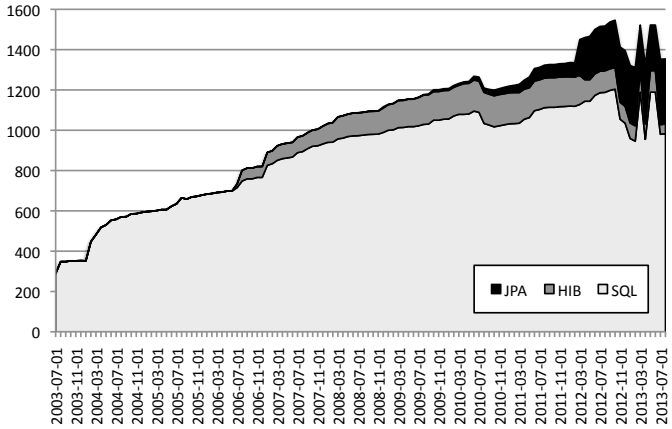

Fig. 5. Monthly evolution of the relative proportion of files of a given type in OSCAR.

JPA-file touches. Between JPA and HIB we did not find any correlation with statistical significance. We conclude that, at the granularity level of file touches, we cannot find a clear separation of responsibilities between database-related and database-unrelated activities.

TABLE II
CORRELATIONS FOR MONTHLY NUMBER OF FILE TOUCHES PER ACTIVITY TYPE. STRONG CORRELATIONS ≥ 0.60 ARE IN **BOLDFACE**. VALUES WITHOUT STATISTICAL SIGNIFICANCE ($p > 0.01$) ARE UNDERLINED.

	SQL	HIB	JPA	pure
SQL		0.53	**0.65**	**0.88**
HIB	0.53		0.15	**0.65**
JPA	**0.65**	0.15		**0.60**
pure	**0.88**	**0.65**	**0.60**	

Research Question 3

With RQ_3, we wish to study how developers distribute their work across code-related and database-related activities, and how the introduction of ORM technology (in particular, Hibernate and JPA persistence providers) affects the database-related activities of developers.

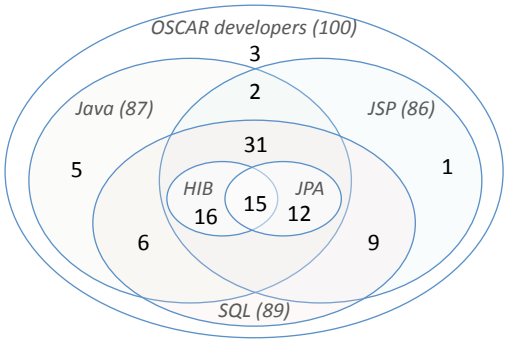

Fig. 6. Number of OSCAR developers involved in file touches per activity type.

To start with, we explore in which types of activity OSCAR developers are involved over the considered lifetime. Fig. 6 shows the types of files touched by each of the 100 distinct developers. 89 of them are involved in activity type *SQL*, which is by construction a subset of *Java* and *JSP*. 76 developers are involved in both Java and JSP, 74 of which are also involved in SQL. Of these, 43 developers are active in HIB and JPA. We therefore confirm our conclusion that there is no separation of responsibilities between the different considered activities.

Fig. 7 visualises the monthly number of file touches developers are involved in over time. We see that developers regularly start contributing to OSCAR, and that in the large majority of cases, the monthly activity of developers is a combination of database-unrelated and database-related file touches (blue colour).

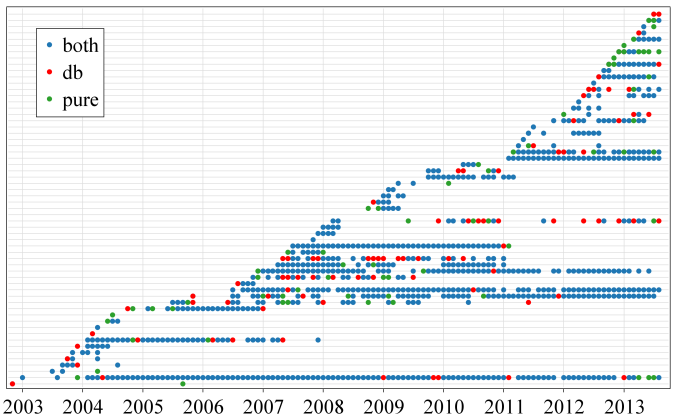

Fig. 7. Evolution of the (monthly) activity type per developer: green = touches to database-unrelated files; red = touches to database-related files; blue = touches to both types of files.

Note that the results reported for RQ_2 and RQ_3 are an overapproximation since we limited ourselves to the granularity level of file touches. For example, a touch to a file containing embedded SQL is counted as part of the activity type *SQL*, even if this touch corresponds to a modification that is unrelated to the database. To compensate for this overapproximation, we also present an underapproximation in the next subsection.

Research Question 4

With RQ_4 we wish to understand who is involved in introducing database-related code. We therefore computed the first commit date at which each code file (regardless of whether it was newly introduced or previously existing) could be related to a database-related activity type (SQL, HIB or JPA). We recomputed the Venn diagrams of Fig. 6 based on this, and present them in Fig. 8. The number of developers involved in *SQL* is now 53 (as opposed to 89 before) and 28 (as opposed to 43 before) are involved in *HIB* and *JPA*.

Fig. 9 shows the monthly aggregated database-related activities per developer on a timescale. We clearly see a transition phase from SQL to HIB and from HIB to JPA. We do not find any new occurrence of HIB after February 2011. On the other

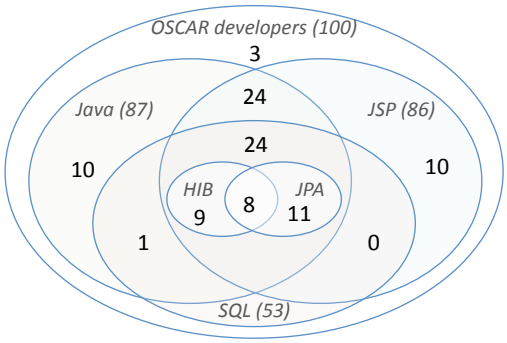

Fig. 8. Number of developers that can be attributed to relating a code file to a database-related activity type (SQL, HIB or JPA) for the first time.

hand, new files continue to be classified as SQL until the end of the analysed period.

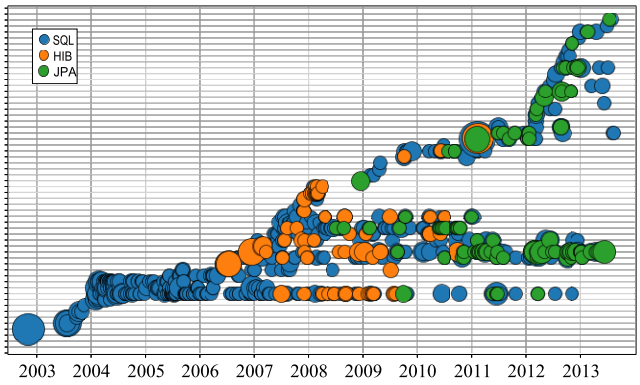

Fig. 9. Number of JSP and Java files per commit date (X-axis) and per developer (Y-axis) that can be newly attributed to SQL, HIB or JPA.

We also computed the Spearman correlations. For SQL versus HIB we obtained $\rho = 0.16$ without statistical significance ($p = 0.13 > 0.01$). For HIB versus JPA we obtained $\rho = 0.09$ without statistical significance ($p = 0.54$). Only for SQL versus JPA we obtained a strong and statistically significant correlation $\rho = 0.89$ ($p < 0.01$).

VI. Threats to Validity and Future Work

The preliminary results reported for OSCAR are not necessarily generalisable to other DISS. We therefore aim to repeat and extend our study by taking into account the systems and research questions by Qui et al. [6].

The main programming language used for OSCAR (i.e., Java) may have played a role in the outcome of the results. We expect to see a difference between statically typed object-oriented languages and dynamically-typed scripting languages designed for web development (e.g., PHP).

Our analysis may be biased by limitations in the Git extraction [10], CVSAnaly2 and measurement tools that we have used. The identity merging we performed may not have been 100% correct either.

Our results are limited by the restriction to the granularity of files and file touches. Analysing the actual impact of changes on the source code would require static program analysis techniques such as the ones used in [5]. Such analysis would be very useful to study the effect of introducing new database technology, as it could allow us to identify particular change patterns associated with the migration towards this new technology. Such change patterns, once identified, could then be encoded into automated refactorings [11] in order to facilitate and automate migration towards the new technology.

VII. Conclusion

The study of data-intensive software systems raises additional research questions and challenges beyond those traditionnally addressed in empirical research. Our study of the co-evolution between code-related and database-related activities in OSCAR aimed to provide preliminary answers to some of these questions. We observed a migration to ORM and persistence technologies offered by Hibernate and JPA, but the practice of using embedded SQL code still remains prevalent today. Contrary to our intuition, we did not find a specialization of developers towards specific database-related activities: the majority of developer appear to be active in both database-unrelated and database-related activities, and this during the entire considered time period. Future studies are needed to find out whether this behavior is also observable in other systems.

Acknowledgment

This research was financed by F.R.S.-FNRS FRFC research project T.0022.13 “Data-Intensive Software System Evolution”, in collaboration with the University of Namur.

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] H. Gall, K. Hajek, and M. Jazayeri, “Detection of logical coupling based on product release history,” in *Int’l Conf. Software Maintenance*, 1998, pp. 190–198.

[5] A. Maule, W. Emmerich, and D. S. Rosenblum, “Impact analysis of database schema changes,” in *Int’l Conf. Software Engineering*, 2008, pp. 451–460.

[6] D. Qiu, B. Li, and Z. Su, “An empirical analysis of the co-evolution of schema and code in database applications,” in *Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. ACM, 2013, pp. 125–135.

[7] 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.

[8] M. Goeminne and T. Mens, “A comparison of identity merge algorithms for software repositories,” *Science of Computer Programming*, vol. 78, no. 8, pp. 971 – 986, 2013.

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

[10] 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’l Working Conf. on Mining Software Repositories*, 2009, pp. 1–10.

[11] J. Kerievsky, *Refactoring to patterns*. Addison-Wesley, 2004.