On the Interaction of Relational Database Access Technologies in Open Source Java Projects

Alexandre Decan*, Mathieu Goeminne*† and Tom Mens*

*Software Engineering Lab, University of Mons, Belgium
Email: { first . last } @ umons.ac.be
†Center of Excellence in Information and Communication Technologies, Belgium
Email: mathieu.goeminne@cetic.be

Abstract

This article presents an empirical study of how the use of relational database access technologies in open source Java projects evolves over time. Our observations may be useful to project managers to make more informed decisions on which technologies to introduce into an existing project and when. We selected 2,457 Java projects on GitHub using the low-level JDBC technology and higher-level object relational mappings such as Hibernate XML configuration files and JPA annotations. At a coarse-grained level, we analysed the probability of introducing such technologies over time, as well as the likelihood that multiple technologies co-occur within the same project. At a fine-grained level, we analysed to which extent these different technologies are used within the same set of project files. We also explored how the introduction of a new database technology in a Java project impacts the use of existing ones. We observed that, contrary to what could have been expected, object-relational mapping technologies do not tend to replace existing ones but rather complement them.

In: A.H. Bagge, T. Mens (eds.): Postproceedings of SATToSE 2015 Seminar on Advanced Techniques and Tools for Software Evolution, University of Mons, Belgium, 6-8 July 2015, published at `http://ceur-ws.org`

1 Introduction

As software systems become more and more complex, the effort required for creating new systems and maintaining existing ones increases over time. This effort can be reduced by embedding code in reusable libraries that offer services for supporting a particular aspect of the developed system. For example, for software systems that strongly interact with a *relational database*, numerous technologies (libraries, APIs and frameworks) exist for connecting the program code to the database. Understanding how database technologies tend to replace or complement existing ones in software projects can help project managers in choosing the most appropriate technology, and the most appropriate moment of introducing this technology.

The program code can be connected to the database in various ways. In the simplest case, the code will contain embedded database queries (e.g., SQL statements) that will be interpreted by the database management system. In more complex cases, especially for object-oriented programs, *object-relational mappings (ORM)* will be provided to translate program concepts (e.g., classes, methods and attributes) into database concepts (e.g., tables, columns and values), so that database elements can be created, read, updated or deleted (CRUD) directly by manipulating object-oriented views. Despite the fact that ORMs abstract away from technical connection details in order to facilitate software development, some evolution-related problems remain.

The high level of dynamic of current database access technologies makes it hard for a programmer to figure out which SQL queries will be executed at a given location of the program source code, or which source code methods actually access a given database table or column. Conversely, the high level of ab-

straction provided by the ORMs makes it hard to determine the impact on the program code of changes in the database schema. In addition, co-evolving the database and the program requires to master multiple languages and technologies.

This paper examines how popular technologies are used in open source Java projects for connecting the source code to a relational database. To do so, we focus on three research questions:

RQ_1 – *When and in which order are database technologies introduced in a project?* We observe that they tend to be introduced very early in the project's lifetime. This is expected, since those technologies are typically central components of the projects in which they occur. We also observe that multiple database access technologies are used in many projects, and that they tend to be used simultaneously. Finally, we study which technologies tend to be complemented by other technologies.

RQ_2 – *How does the introduction of a new technology in a project affect the already included ones?* With this question we wish to understand whether technologies tend to replace existing ones, or rather complement them. In the former case, the introduction of a new technology would decrease the use of the already included technology. In the latter case, the new technology may serve as a catalyst, leading to an increased of the already included technology.

RQ_3 – *To which extent does the introduction of a new technology impact the way in which a project accesses the database?* This question focuses on the evolution of project files that use a particular technology, after introducing a new database technology in the project: are these files modified in order to benefit from the newly introduced technology? For certain pairs of technologies, we found this to be the case. For most pairs of technologies however, existing database-related files do not substantially adopt the latest introduced technology.

The remainder of this paper is structured as follows. Section 2 presents attempts to methodically analyse and compare similar technologies that can be found in the scientific literature and puts our research in perspective. Section 3 presents the approach we followed for collecting the data required for our empirical study as well as the methodology for analysing it. The next three sections address our research questions. Section 7 discusses the threats to validity of our study. Section 8 discusses possible extensions of the presented study, and Section 9 concludes.

2 State of the Art

While the literature on database schema evolution is very large [1], few authors have proposed approaches to systematically observe how developers cope with database evolution in practice. Sjoberg [2] presented a study where the database schema evolution of a large-scale medical application is measured and interpreted. Vassiliadis et al. [3] studied the evolution of individual database tables over time in eight different software systems.

Several researchers have tried to identify, extract and analyse database *usage* in application programs. The purpose of the proposed approaches ranges from error checking [4, 5, 6], over SQL fault localisation [7], to fault diagnosis [8]. More recently, Linares-Vasquez et al. [9] studied how developers document database usage in source code. Their results show that a large proportion of database-accessing methods is completely undocumented.

Several empirical studies have analysed the evolution of library and technology usage. Bauer and Heinemann [10] were able to identify distinct evolution scenarios for API dependencies in software projects. The gained knowledge may be useful for evaluating opportunities in API migration and evolution. Teyton et al. [11] identified sets of similar libraries in a large corpus of software projects. The obtained results can be used for suggesting alternative libraries to project managers who want to migrate from a library to another one. In [12] they investigate how and why library migrations occur. They found that library migrations are relatively rare, and projects that have witnessed more than one migration are exceptional. They also observed that migration is generally an atomic change performed by a single developer in a single commit.

3 Methodology and Data Extraction

The empirical study in this paper focuses on open source Java systems. Java is among the most popular programming languages today, and a large number of technologies and frameworks are available to facilitate relational database access from within Java code. The choice for open source systems is motivated by the accessibility of the entire history of the source code in freely accessible version control repositories.

3.1 Considered Database Access Technologies

In previous work [13, 14], we considered 26 Java relational database technologies that offer a direct means of accessing a relational database and whose presence in a project is identifiable through static analysis. By analysing the import statements in Java files as well as the presence of specific configuration files, we deter-

mined the presence of each of these technologies. We performed a survival analysis of the technologies used in order to determine their relative importance over time in the considered projects.

This paper provides a more in-depth study, by looking at the interaction between object-oriented source code and relational databases at a more fine-grained level. We have selected three popular technologies that are representative of a particular way to connect the source code to a database (embedded SQL, external mapping files, and Java annotations):

JDBC

jdbc[1] is a low-level technology for connecting Java programs to a database by sending SQL queries directly from within the source code. While version 1.1 was released in 1997, there have been regular version upgrades to cope with the evolution of the Java language. This technology is still intensively used in numerous projects [13], despite the inherently close coupling that is required between the source code and the database schema.

In our study we consider this technology as being associated to a Java source code file if entities belonging to `java.sql` are imported in this file.

Hibernate

ORM technologies rely on a mapping description for associating (object-oriented) source code elements to database elements. They aim to reduce the so-called *object-relational impedance mismatch* [15]. The mapping description can take the form of configuration files, placed aside source code files, to express the relations between the considered entities. Hibernate is a popular open source Java framework adopting this solution. It was first released in 2001, and provides an abstraction layer on top of jdbc. Hibernate has been criticised by many of not being a 100% transparent data persistence solution.

In our study we analyse Hibernate[2] XML configuration files (denoted by **hbm** hereafter), and consider that a Java file relies on Hibernate technology if at least one Hibernate configuration file mentions the Java file as a code entity resource.

JPA

Annotation-based mapping descriptions offer an increasingly popular means to express the relations required by ORM engines. With such mappings, Java annotations are used to mark program elements as counterparts of database entities. The *Java Persistence API*[3] (denoted by **jpa** hereafter) is the *de facto* Java standard for annotation-based mappings. jpa was first released in 2006, and relies on the Java annotation mechanism that was first introduced in Java 5. We consider this technology as representative for this kind of mapping description.

In our study we consider that a Java file relates to jpa if the `Entity`, `Embeddable`, or `MappedSuperclass` annotations from package `javax.persistence` can be found in this file.

Discussion

As witnessed by many discussions on Stack Overflow[4], there is no consensus on which of these three technologies is the most appropriate for any given project, as it may depend on many project-related characteristics, technological choices or even personal preferences.

One should also note that the use of these technologies is not exclusive. A project may use all of these technologies simultaneously. These technologies may even be used together within the same Java source code files.

3.2 Selected Projects

In order to obtain a representative project sample, we based our empirical analyses on Java projects belonging the GitHub project corpus proposed by Allamanis and Sutton [16]. Among these projects, 13,307 still had an available Git repository on 24 March 2015.

In order to carry out our empirical study, we selected 2,457 projects from this project corpus for which at least one of the commits contained a reference to either jdbc, jpa or hbm. For each selected project, we extracted the existing relations between source code and database entities from the first commit of each week, and we obtained an historical view of all the files that can be related to a particular technology or to a particular framework.

	mean	stdev	median	max.
duration (in weeks)	76	121	23	812
# commits	1317	6013	126	174,618
# contributors	12	31	4	1091
# files in HEAD	1058	3549	213	103,493
# Java files in HEAD	512	1793	88	46,661

Table 1: Characteristics of the selected projects. HEAD refers to the latest extracted version.

Table 1 shows some of the characteristics of the selected projects. The distribution of metrics values is

[1] `oracle.com/technetwork/java/javase/jdbc/`
[2] `hibernate.org/`
[3] `oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html`
[4] see for example `stackoverflow.com/questions/`Q with Q = 1607819, 2397016, 2560500 or 530215.

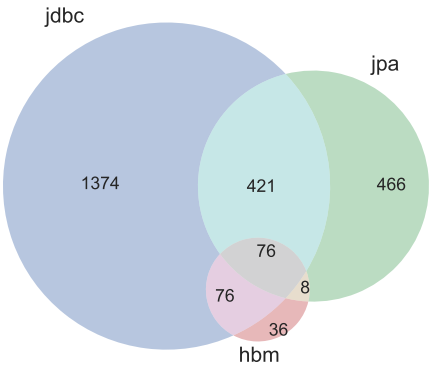

Figure 1: Number of projects per considered technology.

highly skewed, suggesting evidence of a Pareto principle [17]. The duration is expressed in weeks between the first and the last commit.

Figure 1 reports the number of projects per considered technology, taking the entire lifetime of each project into account. We observe that the project sample is relatively unbalanced with respect to the presence of each technology, but each pair of technologies is still represented in a quite a number of projects.

4 RQ_1 When and in which order are database technologies introduced in a project?

Introducing a new technology in a software project comes with a certain cost. A common policy is therefore to introduce such a technology only if the expected benefits outweigh the expected cost.

For each project, we analysed at what moment in the projects' lifetime each considered technology got introduced. The answer appears to depend on the duration of the considered projects. To minimise the effect of project duration, we normalised the lifetime of each project into a range between 0 (the start of the project) and 1 (the last considered commit).

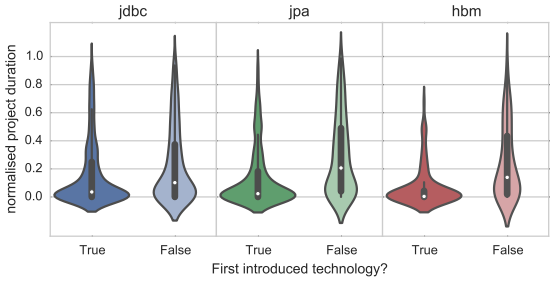

Figure 2: Violin plot (using a kernel density estimate) of the distribution of the introduction time of a technology in the Java project corpus.

Figure 2 compares, for each considered technology, two distributions of the introduction time of the technology in a project. The first distribution (left) considers the first time a technology gets introduced in a project. The second distribution (right) considers the introduction of the technology in a project that already had a technology before. As expected, we observe that **more than 50% of the introductions of a first technology are done in the first 10% of the project's lifetime**. For technologies introduced after an existing one, the distribution tends to be flatter.

We also observe that the two distributions for jdbc present less differences than the ones related to jpa or hbm. To achieve this, we performed a Kolmogorov-Smirnov statistical test for each pair of distributions related to jdbc, jpa and hbm. The tests show that **the two distributions associated to each technology are significantly different** (p-values are lower than 10^{-6}). This may indicate that **for jdbc, the moment of introduction is less affected by the presence of another technology than for hbm and jpa**.

We saw that the time at which a technology is introduced in a project varies depending on the presence of another technology in this project. What are the technologies that are more likely to be succeeded by another one?

To answer this question, we use the statistical technique of *survival analysis* to estimate the probability that a technology does not remain the last introduced one in a project lifetime. Survival analysis [18] creates a model estimating the survival rate of a population over time, considering the fact that some elements of the population may leave the study, and for some other elements the event of interest does not occur during the observation period. In our case, the observed event is the introduction in a project of another technology after an existing one.

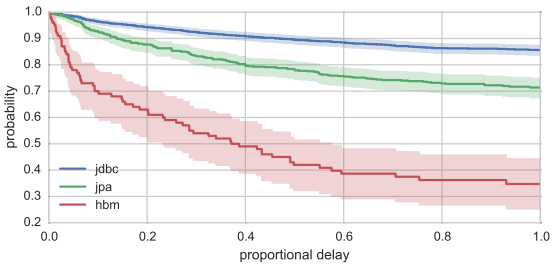

Figure 3: Probability that a technology remains the last introduced technology over time.

Figure 3 shows the survival rates for each considered technology. We observe that hbm has a much lower survival rate (i.e., a lower probability of staying the last introduced technology for a long time) than the other technologies. We also observe that, during the first 10% of the projects' lifetime, the survival rates

of hbm decrease by 30%, representing a more important decrease than for the other two technologies. This implies that **hbm is usually quickly replaced or complemented by another technology**.

Figure 1 showed that around 23% of the projects use two or more database technologies in their lifetime, but these are not necessarily used simultaneously. We therefore identified which combinations of technologies actually *co-occur* in the selected Java projects. Frequent co-occurrences would reveal which technologies are complementary, and which technologies are used as supporting technologies of other ones. For each pair of technologies, we counted the number of projects in which these technologies actually co-occur, and in which order they were introduced in these projects. The results are summarised in Table 2.

$(A, B) \rightarrow$	(jdbc, jpa)	(jdbc, hbm)	(jpa, hbm)
# projects	497	152	84
# co-occurrences	488	148	77
% co-occurrences	98.2%	97.4%	91.7%
$\text{start}_A < \text{start}_B$	157	50	19
$\text{start}_A > \text{start}_B$	151	27	37
$\text{start}_A = \text{start}_B$	189	75	28

Table 2: Projects characteristics by pairs (A, B) of co-occurring technologies

Among all projects that use multiple technologies during their lifetime we observe a very high proportion of co-occurring technologies. More specifically, in 97.3% (488+148+77 out of 497+152+84) of all the situations in which two distinct technologies were used during a project's lifetime, they were used simultaneously. Around 41% (189+75+28 out of 488+148+77) of all pairs of co-occurring technologies were introduced simultaneously ($start_A = start_B$), implying that around 59% of all pairs of co-occurring technologies concern projects in which the technologies were introduced at different moments ($start_A \neq start_B$).

Considering the number of projects in which the introduction of a technology A was observed before the use of a technology B, it seems that **jpa tends to succeed to hbm more often than the contrary** (37 versus 19 observations). Similarly, **hbm tends to succeed to jdbc more often than the contrary** (50 versus 27 observations). We did not identify such an order for jpa and jdbc (151 versus 157 observations).

Summary. All considered technologies are introduced early in the projects' lifetimes, even for projects that already use another technology. The number of projects in which multiple technologies co-occur is proportionally important. The order in which these technologies are introduced suggests that hbm is often succeeded by jdbc or jpa.

5 RQ_2 How does the introduction of a new technology in a project affect the already included ones?

As multiple database access technologies are used in many projects, either simultaneously or one after the other, it is useful to study how the introduction of a new technology can impact the use of an already included one. This impact, if it occurs, could result in an increased or decreased usage of the already included technology. We therefore identified and counted for which projects the introduction of a new technology causes an increasing use of the older technology, a decreasing use, or no observable change in the use of the already included technology.

To qualify the impact, we rely on the *first derivative* of the number of files related to an existing technology. We computed and compared the mean of this derivative for two 8-week periods: the first period strictly precedes the moment of introduction of the new technology, and the second period immediately follows the moment of introduction.

In the following, we will use the term *variation* to denote the difference between the mean of the second period and the mean of the first period. The variation of a technology is easy to interpret: a positive value indicates an increasing use of the existing technology while a negative value indicates a decreasing use of the existing technology

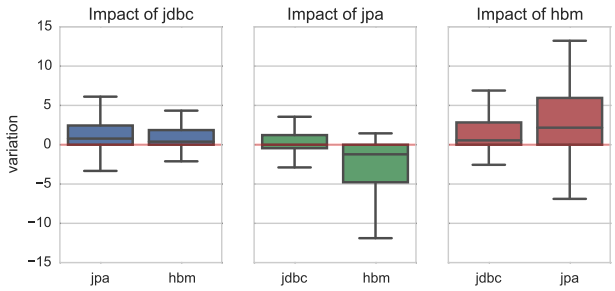

Figure 4: Impact of the introduction of a new technology on the activity of an already included technology.

Figure 4 shows the distribution of the variation for each pair of technologies. We observe that **jdbc and hbm cause a slight positive impact on the use of existing technologies** (since the variation tends to be positive in 75% of all cases). Notice the important variation induced by introducing hbm in projects using jpa. The converse is not true: **introducing jpa in a project that already uses hbm implies a negative variation for hbm**.

Figure 4 only identifies global trends in our project corpus. It does not allow to identify trends within individual projects. Figure 5 therefore distinguishes the projects that exhibit a positive variation (**blue**

curve), a negative variation (**red curve**) or no variation (**green curve**) for several time intervals after the introduction of the new technology.

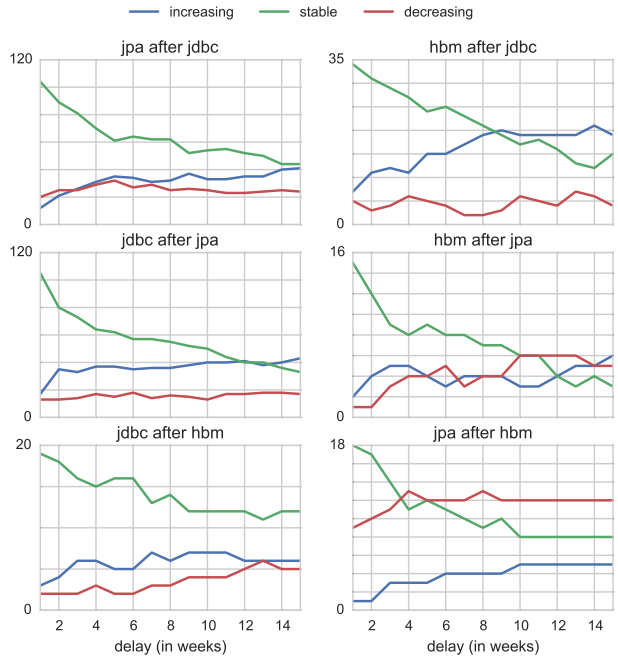

Figure 5: Number of projects with an increasing, decreasing or stable activity of an already included technology, as observed x weeks after introducing another technology.

Regardless of the considered pair of technologies, with the notable exception of the pairs (jpa after hbm) and (hbm after jpa), both the number of projects having no variation and the number of projects having a positive variation are systematically greater than the number of projects exhibiting a negative variation.

Figure 6 shows survival curves, using a Kaplan-Meier estimator, of the probability that a project keeps more than a threshold of 25% of its files related to an already included technology after the introduction of new one. We tried different threshold values and they all lead to the same conclusions.

Again, we observe that **the most distinct behaviours are exhibited by jpa and hbm**: the probability to keep more than 25% of files related to hbm drops below 0.55 about 20 weeks after introducing jpa, while the probability for jpa files drops to a little more than 0.6 about 19 weeks after introducing hbm. This analysis corroborates our previous observations: **introducing jpa or hbm does not negatively impact the use of jdbc**, and conversely. We also observe from Figures 5 and 6 that **most of the impact happens in the first weeks after introducing the new technology**.

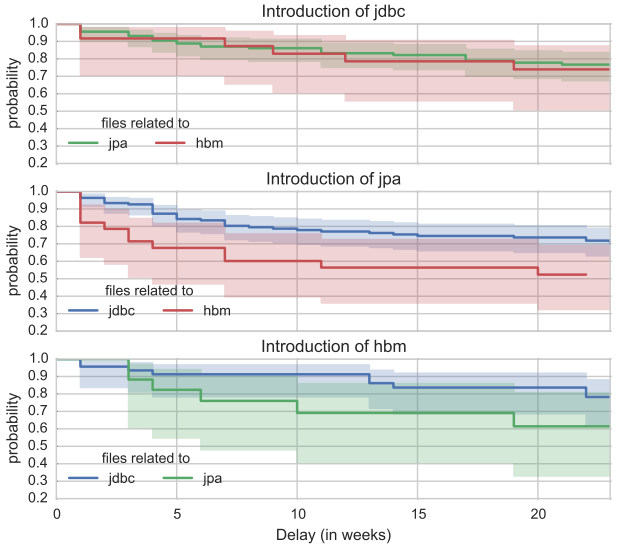

Figure 6: Probability that at least 25% of files related to a technology remain after the introduction of another technology.

Summary. Introducing a new technology generally induces, in the short term, an increase of the presence of the already included technology, with the notable exception of the introduction of jpa on a project that already makes use of hbm. This suggests that, contrary to the promises of ORM technologies, new technologies do not tend to replace existing ones but rather complement them.

6 RQ_3 To which extent does the introduction of a technology impact the way in which a project accesses the database?

From the results of RQ_1 we observed that, if a project uses multiple database access technologies over its lifetime, these technologies tend to co-occur. At a more fine-grained level, we are interested in the impact of the introduction of a technology on the files that already relate to a previously used technology.

6.1 Do different technologies co-occur at file level?

Let us first study the co-occurrences of different technologies at file level without taking the evolutionary aspect into account. Figure 7 shows, for each pair of technologies, the distribution across projects of the ratio between the number of files that relate to each, or both, technologies, and the number of files that relate to any of these technologies. For each pair of tech-

nologies, only projects in which both technologies have been used at some point in their lifetime have been retained as elements of the distribution.

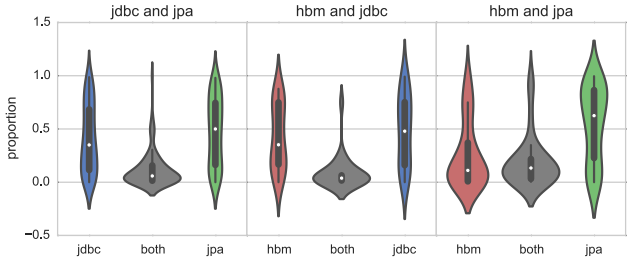

Figure 7: Relative number of files relating to pairs of technologies.

It turns out that pairs of technologies including jdbc present similar profiles: most projects contain a small proportion of files using both technologies. A two-sided Kolmogorov-Smirnov test confirms this similarity between distributions: we cannot reject the null hypothesis that states that the distributions associated to the proportion of files using a single technology are identical ($p = 0.877$ and 0.287, respectively). We conclude that **jdbc is generally not used in the same files as jpa and hbm**.

The pair of technologies jpa and hbm presents a different behaviour. The three distributions of the proportion of files that only relate to these technologies are significantly different (we reject the null hypothesis with $p < 0.001$). This result, combined with the form of the distributions, suggests that, for projects having used jpa and hbm, **a file is likely to relate either to jpa only or to both jpa and hbm**. In addition to this, the proportion of files that use both hbm and jpa is more important than for the other considered pairs of technologies.

Summary. There is a clear separation between files using jdbc and files using the two other technologies. For the combination of hbm and jpa, a partial, asymmetric overlap exists at file level: hbm is often used in the same files as jpa, while jpa is rarely used in combination with another technology in the same file.

6.2 How does the co-occurrence of technologies at file level evolve over time?

Let us now look at the same question from an evolutionary point of view, by assessing the impact, at file-level, of introducing a new technology in a project that already uses another technology to access the database. To do this, we study how the files related to an existing technology get changed after introduction of the new technology.

Let us associate a *migration profile* to each project at different points in time after the introduction of the new technology. This migration profile reflects how the files related to the old technology are impacted. It is computed as follows:

Let P be a project and $\mathcal{T} = \{\mathsf{jdbc}, \mathsf{hbm}, \mathsf{jpa}\}$ the considered technologies. For each point in time t for P and each technology $A \in \mathcal{T}$ we define $\mathsf{related}_P(A, t)$ as the (possibly empty) subset of (fully qualified) filenames of P in which technology A was detected at time t.

For every pair of distinct technologies $(A, B) \in \mathcal{T} \times \mathcal{T}$, we write $M = (P, A, B)$ if P is a project in which technology B gets introduced while a technology A is already in use. Let t_M denote the point in time of this introduction and $F_M = \mathsf{related}_P(A, t_M)$ the set of filenames associated to technology A. For each $t \geq t_M$ we associate to each $f \in F_M$ a label in $\mathcal{L} = \{\mathit{residual}, \mathit{removed}, \mathit{complemented}, \mathit{replaced}\}$ as follows:

residual if $f \in \mathsf{related}_P(A, t) \setminus \mathsf{related}_P(B, t)$
removed if $f \notin \mathsf{related}_P(A, t) \cup \mathsf{related}_P(B, t)$
complemented if $f \in \mathsf{related}_P(A, t) \cap \mathsf{related}_P(B, t)$
replaced if $f \in \mathsf{related}_P(B, t) \setminus \mathsf{related}_P(A, t)$

Given M, we also associate to each $t \geq t_M$ a set of labels $\mathsf{mp}_\mathsf{M}(t) \subseteq \mathcal{L}$. A label $L \in \mathcal{L}$ belongs to $\mathsf{mp}_\mathsf{M}(t)$ if, among the labels associated to each $f \in F_M$ at time t, no other label occurs more frequently than L.

Finally, the *migration profile* of M at time t is a unique label from $\mathsf{mp}_\mathsf{M}(t)$ selected based on the total order *replaced* > *complemented* > *removed* > *residual*. This total order privileges migration profiles that correspond to the adoption of the new technology.

As the choice of a total order could have altered the results of our analysis, we compared the results obtained with several total orders, and we observed only slight local variations. This is not surprising as there are only 72 pairs (M, t) such that $|\mathsf{mp}_\mathsf{M}(t)| > 1$, representing 1.78% of all the considered pairs.

Figure 8 shows the evolution of the proportion of projects with a given migration profile. For the sake of readability, we only present results for *complemented*, *replaced*, and *removed*. The results for *residual* can be deduced from these, by taking the complement of *complemented*, *replaced* and *removed*.

We observe that, for each considered pair of technologies, and for each time delay (expressed in weeks) after the introduction of the new technology, most projects relate to the *residual* migration profile, implying that projects tend not to adapt their existing database access files to make use of the newly introduced technology. This is especially true for projects introducing jdbc after jpa or hbm.

The second dominant migration profile is *removed*. Regardless of the considered pair of technologies, more

Figure 8: Proportion (stacked) of projects for each migration profile. The complement corresponds to *replaced*.

and more projects are associated to this migration profile. Over time, an increasing number of projects tend to reduce the number of files relating to the first considered technology. The predominance of *residual* and *removed* migration profiles seems to convey that, **in many cases, files that related to the existing technology are not prone to use the newly introduced technology.** Instead, they either continue to use the first technology or they tend to lose any relation to database access management.

The two other migration profiles, *complemented* and *replaced*, indicate an effective file migration from the existing technology to the newly introduced one. Such cases appear to be much less represented in our corpus, with the exception of projects in which jpa or jdbc is introduced after hbm. This is especially the case **when jpa is introduced in a project using hbm: the files that were related to hbm become (sometimes exclusively) related to jpa.**

Summary. Different technologies generally do not tend to co-occur in the same set of files, except, to some extent, when jpa and hbm are used together. We do not observe a true migration in technology usage: files that are related to a given technology do not tend to adopt the newly introduced technology, except for projects that migrate from hbm to another technology.

7 Threats to validity

Our research suffers from the same threats as other research relying on Git and GitHub [19, 20].

The selected Java projects potentially suffer from the same generalisability constraints as in [16]. The open source GitHub Java project corpus was curated to exclude low-quality projects (by ignoring projects that were never forked) and project duplicates.

While our corpus contained 2,457 projects, the number of projects involved in some pairs of database technologies were sometimes much lower. For example, only 19 projects were concerned by a migration from jpa to hbm (cf. Table 2). The accuracy of our observations could be increased by using a larger project corpus.

The detection of a technology is based on the static analysis of code and project-specific artefacts (e.g., Java annotations, import statements and XML files). This approach can lead to false positives: the presence of these artefacts does not necessarily reflect the actual use of the related technology.

Some of our analyses are based on arbitrarily chosen thresholds and on weekly time intervals. Because our results may depend on these thresholds and intervals, we repeated our experiments with different parameters but did not observe any major differences.

8 Future Work

The results presented in this article, possibly combined with more traditional project quality metrics, could be integrated in a managerial dashboard. Such a dashboard could be used to compare the characteristics and the evolution of a particular project against those belonging to the analysed project corpus. This would support project managers in evaluating and exploiting the expected benefits and disadvantages from introducing a new technology, as well as in assessing the impact of how this technology will become used in the project over time. Any ensuing managerial decisions will obviously depend on project-specific rules and guidelines that could hardly be generalized.

This paper used static analysis techniques to detect the presence of a particular technology. Using dynamic analysis techniques could reveal how database technologies are actually used in running systems. The analysis of queries submitted to the database at runtime could be used for understanding to which extent ORM technologies hide complexity to developers.

This paper focused on relational database access technologies based on three representative technologies (jdbc, Hibernate and jpa). It could be useful to include other Java specifications for object persistence as well, such as JDO. It would also be useful to consider other

kinds of databases (such as NoSQL, graph or object-oriented databases), since these are becoming increasingly more popular. A follow-up study could take into account such alternative database technologies.

Other technological domains (beyond databases) could be considered as well. Event loggers, graphical user interfaces, and unit tests are examples of features supported by multiple concurrent technologies. Since the identification of the technology used in project files is the only part of our methodology that depends on the considered technologies, our approach could be easily adapted to study other technologies.

Section 7 mentioned the limitations of the selected project corpus. We therefore intend to confirm our research results by considering a larger project corpus, including both open and closed source projects. We also intend to study the effect of project quality and project maturity on the obtained results. Finally, we intend to include other programming languages than Java in the project corpus in order to avoid any bias introduced by language-specific characteristics.

While this paper only focused on *technical* aspects of connecting source code to databases, we plan to study the *social* aspects of systems involving such a database connection. More precisely, we would like to determine if the different technologies are introduced and managed by different teams or persons. Inspired by [21] we also aim to analyse the developer characteristics in order to determine how these affect the take-up, use, evolution and migration of technologies. Some examples of developer characteristics are their degree of specialisation, diversity, seniority, skills, and workload.

Finally, we plan to analyse software systems in order to automatically identify library features used in the source code, as well as feature similarities between different technologies. In situations where developers want to migrate from a given technology to another, such a feature identification and mapping is a first step towards better support for assisted or automatic migration [22].

9 Conclusions

Through static analysis of Java source code we carried out a large-scale empirical study to understand how database access technologies interact with one another. We considered three popular technologies (jdbc, hbm and jpa) that represent different means to connect Java source code files to a relational database. We selected data from 2,457 open source projects on GitHub that used at least one of the considered technologies.

Our study revealed common behaviours in the use of these three technologies. In spite of the promises of ORM technologies, we found no evidence that the low-level jdbc solution is massively replaced by hbm or jpa. The only significant technology migration we observed concerns the transition from hbm to jpa. More specifically, we summarise our main observations below.

We analysed the evolution and co-occurrences of the technologies in order to get a high-level view of their usage in the considered Java projects. It appears that, most of the time, database technologies are introduced early in the projects' lifetime, whether they are the first technology introduced or not. Once introduced in a project, hbm tends to be complemented or replaced by another technology more frequently and more quickly than jpa and jdbc.

We also analysed how the technologies are used in the source code files. The introduction of jdbc and hbm tends to be followed by an increasing use of the already present database technology. This increase is particularly important when hbm is introduced after jpa. Conversely, the introduction of jpa reduces the use of hbm. jpa therefore appears to replace existing hbm in the database-related source-code files, while the converse is not true.

Furthermore, jdbc generally does not share source code files with the two other considered database technologies. While jpa is used in isolation in a majority of source code files, hbm tends to be used more often in conjunction with jpa. The study of the evolution of such co-occurrence reveals that a file migration from a technology to another one is only observed from hbm to jpa. In most projects, the introduction of a new database technology is not followed by a massive adoption of this technology by the existing database-related files, until these files become database-unrelated or are removed from the source code repository.

Exploiting all these results in a dashboard that supports managers in making project-specific decisions with respect to the introduction, use or evolution of database access technologies remains part of future work.

Acknowledgment

This research was conduced as part of the FRFC research project T.0022.13 "Data-Intensive Software System Evolution" that was financed by the F.R.S.-FNRS, Belgium.

References

[1] E. Rahm and P. A. Bernstein, "An online bibliography on schema evolution," *SIGMOD Rec.*, vol. 35, no. 4, pp. 30–31, Dec. 2006.

[2] D. Sjoberg, "Quantifying schema evolution," *Information and Software Technology*, vol. 35, no. 1, pp. 35 – 44, 1993.

[3] P. Vassiliadis, A. V. Zarras, and I. Skoulis, "How is life for a table in an evolving relational schema? Birth, death and everything in between," in *Int'l Conf. Conceptual Modeling (ER)*, 2015, pp. 453–466.

[4] A. S. Christensen, A. Møller, and M. I. Schwartzbach, "Precise analysis of string expressions," in *Int'l Conf. Static Analysis (SAS)*, 2003, pp. 1–18.

[5] C. Gould, Z. Su, and P. Devanbu, "Static checking of dynamically generated queries in database applications," in *Int'l Conf. Software Engineering*. IEEE Comp. Soc., 2004, pp. 645–654.

[6] M. Sonoda, T. Matsuda, D. Koizumi, and S. Hirasawa, "On automatic detection of SQL injection attacks by the feature extraction of the single character," in *Int'l Conf. Security of Information and Networks (SIN)*, 2011, pp. 81–86.

[7] S. R. Clark, J. Cobb, G. M. Kapfhammer, J. A. Jones, and M. J. Harrold, "Localizing SQL faults in database applications," in *Int'l Conf. Automated Software Engineering (ASE)*, 2011, pp. 213–222.

[8] M. A. Javid and S. M. Embury, "Diagnosing faults in embedded queries in database applications," in *EDBT/ICDT'12 Workshops*, 2012, pp. 239–244.

[9] M. Linares-Vasquez, B. Li, C. Vendome, and D. Poshyvanyk, "How do developers document database usages in source code?" in *Int'l Conf. Automated Software Engineering (ASE)*, 2015.

[10] V. Bauer and L. Heinemann, "Understanding API usage to support informed decision making in software maintenance," in *European Conf. Software Maintenance and Reengineering*, 2012, pp. 435–440.

[11] C. Teyton, J. Falleri, and X. Blanc, "Mining library migration graphs," in *Working Conf. Reverse Engineering*, 2012, pp. 289–298.

[12] C. Teyton, J. Falleri, M. Palyart, and X. Blanc, "A study of library migrations in Java," *Journal of Software: Evolution and Process*, vol. 26, no. 11, pp. 1030–1052, 2014.

[13] M. Goeminne and T. Mens, "Towards a survival analysis of database framework usage in Java projects," in *Int'l Conf. Software Maintenance and Evolution*, 2015.

[14] M. Goeminne, A. Decan, and T. Mens, "Co-evolving code-related and database-related changes in a data-intensive software system," in *CSMR-WCRE Software Evolution Week*, 2014, pp. 353–357.

[15] M. N. C. Ireland, D. Bowers and K. Waugh, "A classification of object-relational impedance mismatch," in *Intl Conf. Advances in Databases, Knowledge, and Data Applications (DBKDA)*, 2009, pp. 36–43.

[16] M. Allamanis and C. Sutton, "Mining source code repositories at massive scale using language modeling," in *Int'l Conf. Mining Software Repositories*. IEEE, 2013, pp. 207–216.

[17] M. Goeminne and T. Mens, "Evidence for the Pareto principle in open source software activity," in *Workshop on Software Quality and Maintainability (SQM)*, ser. CEUR Workshop Proceedings, vol. 701. CEUR-WS.org, 2011, pp. 74–82.

[18] I. Samoladas, L. Angelis, and I. Stamelos, "Survival analysis on the duration of open source projects," *Information & Software Technology*, vol. 52, no. 9, pp. 902–922, 2010.

[19] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. Germán, and P. T. Devanbu, "The promises and perils of mining Git," in *Int'l Conf. Mining Software Repositories*, 2009, pp. 1–10.

[20] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. Germán, and D. Damian, "The promises and perils of mining GitHub," in *Int'l Conf. Mining Software Repositories*, 2014, pp. 92–101.

[21] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload: A case study of the Gnome ecosystem community," *J. Empirical Software Engineering*, pp. 1–54, 2013.

[22] C. Teyton, J.-R. Falleri, and X. Blanc, "Automatic discovery of function mappings between similar libraries," in *Working Conf. Reverse Engineering*, Oct 2013, pp. 192–201.