Data & Knowledge Engineering 98 (2015) 54–73

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Improving conceptual data models through iterative development

Tilmann Zäschke [a,*], Stefania Leone [b], Tobias Gmünder [a], Moira C. Norrie [a]

[a] Institute for Information Systems, ETH Zurich CH-8092 Zurich, Switzerland
[b] Semantic Information Research Laboratory, CS Department, USC Los Angeles, CA, 90089-0781, USA

ARTICLE INFO

Available online 10 July 2015

Keywords:
Database profiling
Evolvability
Object database
Agile development
Conceptual models
Model quality
Semantic verification

ABSTRACT

Agile methods promote iterative development with short cycles, where user feedback from the previous iteration is used to refactor and improve the current version. To facilitate agile development of information systems, this paper offers three contributions. First, we introduce the concept of *evolvability* as a model quality characteristic. Evolvability refers to the expected implications of future model refactorings, both in terms of complexity of the required database evolution algorithm and in terms of the expected volume of data to evolve. Second, we propose extending the agile development cycle by using database profiling information to suggest adaptations to the conceptual model to improve performance. For every software release, the database profiler identifies and analyses navigational access patterns, and proposes model optimisations based on data characteristics, access patterns and a cost–benefit model. Based on an experimental evaluation of the profiler we discuss why the quality of conceptual models can generally benefit from profiling and how performance measurements convey semantic information. Third, we discuss the flow of semantic information when developing and using information systems.
Beyond these contributions, we also make a case for using object databases in agile development environments. However, most of the presented concepts are also applicable to other database paradigms.

1. Introduction

Agile methods [1] are now widely accepted in software engineering. For the development of information systems, agile practices of early adoption and frequent software releases mean that, from the first release on, depending on the project settings, users may populate the database with application-specific data, see [2] for an example. One aim of our work is to exploit the presence of such real-world data at development time to optimise and verify the conceptual data model with respect to database performance. We therefore propose an approach that complements the feedback loop inherent to agile development practices with recommendations for conceptual model optimisations based on database profiling information.

Another aim of our work is to improve model quality beyond database performance. While model optimisation is traditionally applied to the physical or logical models of relational databases, we show how the conceptual model of an application can be optimised, resulting not only in faster applications but also in better models in terms of model quality and correctness. In terms of model quality, we also introduce a novel quality that describes how fast and easy a model and the corresponding database schema and data can be evolved. This notion of 'evolvability' is clearly useful in agile projects where schema and data evolution is frequent.

* Corresponding author.
E-mail addresses: zaeschke@inf.ethz.ch (T. Zäschke), stefania.leone@usc.edu (S. Leone), norrie@inf.ethz.ch (M.C. Norrie).

http://dx.doi.org/10.1016/j.datak.2015.07.005
0169-023X/

We use two main approaches to support the design and refactoring of data models while ensuring model quality. Our first approach uses standard static model analysis which, as the name suggests, is based on a static analysis of the application data model. Since static model analysis is inherently limited to the information specified in the model, it can only give a rough guidance for improving the quality of the conceptual model. However, while it cannot benefit from actual database usage statistics, it can still serve as a useful tool during the data model design process because the results are available in the model editor immediately after any model editing operation.

Our second approach to model optimisation is database profiling. Although it is traditionally advocated that conceptual models should be independent of the underlying database materialisation, we argue that database profiling can be exploited to improve the quality of a conceptual model based on insights into how a database is used and the data accessed. In addition to highlighting possible performance issues, it can also indicate whether the actual application usage correlates with the intended usage of the conceptual model. Comparing actual and intended usage provides opportunities for adapting and improving the model in accordance with evolving requirements and verifying whether the specified requirements match the actual usage. While it is already good practice to involve the end-users in the evaluation of the quality of conceptual models, database profiling has the advantage that it requires no additional effort from their side.

Using the profiling information, we can construct recommendations for refactorings in the data model. In this publication, we demonstrate the effectiveness of profiling and subsequently recommended refactorings. Based on the results from an example scenario, we argue that such refactorings will not only improve the application performance, but can also indicate and resolve semantic problems in the conceptual layer.

Finally, we take this thought a step further and discuss more generally the flow of semantic information when developers and database administrators (DBAs) construct an information system from user requirements and when this information system is eventually used by users. This results in some interesting considerations, for example that mapping a UML model to a relational data model and subsequent normalisation may effectively be seen as a weakness of the development process because it prunes the database data model of certain semantic information, such as inheritance information, and subjective understanding of the problem domain. However, our reasoning suggests that semantic information and subjective understanding are crucial for improving the quality of the conceptual model through profiling.

We focus here on object databases (ODBMSs) as the case has been made that they are well-suited to agile development [3] since they simplify the model and database evolution process. We support this view based on our own experience with a multi-terabyte real-world ODBMS [4]. The simplification of the development process follows from the fact that, in object databases, the conceptual model corresponds to the logical model and, unlike relational databases, there is no need for an object-relational-mapping layer. Changes to the conceptual model between frequent software releases require therefore only the change of the conceptual model, whereas relational databases (RDBMSs) also require adaptation of a separate logical model and the mapping layer between conceptual and logical model.

ODBMSs are especially well-suited to profiling of conceptual models because, in comparison, profiling the logical model in an RDBMS requires a reverse-mapping of the profiling results, which may not be trivial. Furthermore, we expect that profiling results from RDBMSs tend to be semantically less rich, because the process of normalisation in relational database models may remove semantic information.

In contrast to relational databases, ODBMSs support data access through navigation as well as queries. While query optimisation is typically supported through additional data structures such as indexes, navigation is reference-based and therefore tightly bound to the conceptual model. Our approach focuses on model optimisations based on profiling navigational data access paths. For every agile software release, we track and analyse the data access paths and data characteristics in order to propose conceptual model optimisations based on model refactoring patterns and evaluated with a cost–benefit model. The recommendations can be complemented with recommendations from query and static model analysis. We implemented our profiling framework as part of the ZooDB[1] object database and integrated it into AgileIS, a framework for model-driven agile information system development [4]. Recommendations for model optimisations resulting from database profiling are visualised as part of the graphical model editor. As proof of concept, we developed a research publication management system with data from DBLP [5] using our agile development framework together with the extended ZooDB and measured the performance improvements resulting from the recommended optimisations.

The contributions of this publication can be summarised as follows:

- We present the idea of *evolvability*.
- We demonstrate the concept of profiling conceptual models.
- We discuss the flow of semantic information through database usage.

This article extends [6] to show not only how agile development practices can be improved to include conceptual modelling and how object databases support agile development practices, but also what profiling can contribute to model quality beyond model performance and how evolvability can be derived from model analysis.

In Section 2, we discuss related research and then present our approach in Section 3. This is followed by a more detailed discussion of static model analysis in Section 4. Section 5 introduces database profiling with Section 5.1 discussing navigational access paths, while the refactoring patterns and associated cost–benefit models are presented in Section 5.2. In Sections 5.3 and 5.4, we detail the collection of profiling data and how it is brought to the developer in a model-driven development environment. A case study

[1] https://github.com/tzaeschke/zoodb.