An Empirical Study on the Use of SQL Trace Visualization for Program Understanding

Nesrine Noughi
PReCISE Research Center
University of Namur
nesrine.noughi@unamur.be

Stefan Hanenberg
paluno - The Ruhr Institute for Software Technology
University of Duisburg-Essen
stefan.hanenberg@uni-due.de

Anthony Cleve
PReCISE Research Center
University of Namur
anthony.cleve@unamur.be

1 Introduction

Software maintenance and evolution processes often require a sufficient understanding of the current software system version, before the latter can be adapted to new or changing requirements. Several program comprehension techniques have been proposed to assist developers in this preliminary phase such as static program analysis [YPZ09, DG03], dynamic program analysis [ACD09, CH08, CMH11, CNH12] and visualization [LD02, CZvDvR09, JSB].

Nevertheless, those techniques generally fail in producing meaningful behavioral models in the case of modern data-intensive systems — systems that access a large quantity of data in order to support user activities in different contexts. Such systems access an increasing amount of data usually stored in a database. They are characterized by complex, dynamic and continuous interactions between the application programs and their database. Hence the need for program comprehension techniques that are suitable for analyzing this aspect of the programs behavior.

As a modest contribution to this field, we recently proposed an approach aimed at supporting the comprehension of *data-intensive programs*, by focusing on their *database manipulation behavior* [NMMC14]. The approach combines the use of dynamic analysis and visualization techniques to capture, analyze and visualize the interactions between a program and its database under given execution scenarios. The approach is implemented into an integrated tool, called DAViS (Dynamic Analysis and Visualization of SQL execution traces). DAViS, designed for programs relying on a relational database, allows developers to answer questions such as (1) *Which SQL queries are executed by this scenario*; (2) *Which database schema elements (tables, columns) are accessed by these queries?*; and (3) *How do the successive SQL queries depend on each other?*.

Here, we introduce an experiment that empirically assess the usefulness of DAViS for program understanding, and in particular for database manipulation behavior (DBM) comprehension. We also investigate which types of DMB comprehension tasks benefit most from the trace visualization support provided by DAViS. To address these questions, the experiment quantitatively measures how DAViS influences (1) the time required to achieve typical DBM comprehension tasks, and (2) the correctness of the answers related to those comprehension tasks. It turned out that DAViS achieves an improvement in regard to both measurements: the response time is reduced and the correctness is increased.

2 Used Scenario

We evaluate our approach in the context of a Learning Management Systems (LMS) that allows the administration to manage all about teachers, teaching, students and courses, etc.

This system, called AcadYearManager, consists of thousands of lines code written in Java. It manipulates a MySQL database consisting of 30 tables and 192 columns representing the data on available courses, faculties,

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

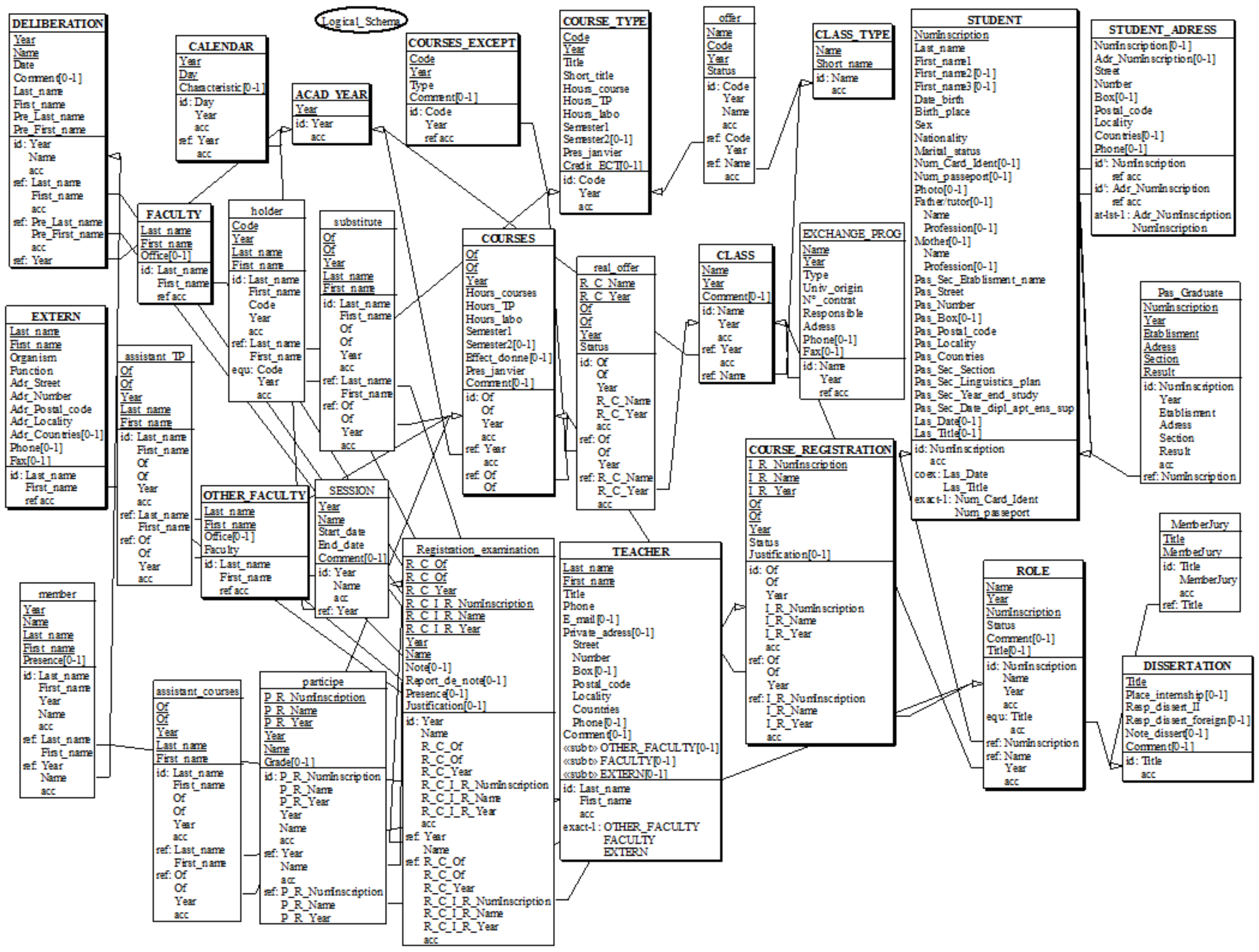

Figure 1: Physical Schema of AcadYearManager

Table 1: Characteristics of the two scenarios under study

Scenario	Number of queries	Types of queries
Scenario 2	60	Select queries
Scenario 3	65	Select queries

teachers, students, classes, etc. Figure 1 depicts the database schema of the AcadYearManager application.

The first step that we performed on the AcadYearManager system was to collect SQL execution traces corresponding to two typical program execution scenarios. For each scenario, we recorded an event log file with the SQL queries executed during the scenario. For each executed query in the trace, we also recorded its result. Each collected execution trace contains a set of SQL queries, each belonging to one of the two different scenarios. Table 1 summarizes the content the two SQL execution traces. Figure 2 shows 4 SQL queries belonging to the execution trace of *"scenario 2"*.

3 Research Questions and Hypotheses

The goal of the experiment is to reflect on two research questions related to the use of DAViS:

- RQ_1: *Does DAViS reduce the time needed to complete the tasks?*
- RQ_2: *Does DAViS increase the correctness of the tasks?*

In order to address these research questions in an experiment, we need to transform them into testable hypotheses:

```
...
SELECT DISTINCT (`Of__Code`),  `Hours_courses` FROM academic.`courses`;
SELECT * FROM academic.`assistant_courses` WHERE `Of__Code`= `infob12`;
SELECT `Name` FROM  academic.`member` WHERE  `Name` LIKE  `%2015%`;
SELECT `Name`, `Year` FROM academic.`exchange_prog`;
...
```

Figure 2: Overview of part of the scenario 2

- $H0_1$: *There is no difference in the response time for different tasks between participants using DAViS and those ones that do not use DAViS.*
- $H0_2$: *There is no difference in the correctness of responses between participants using DAViS and those ones that do not use DAViS.*

References

[ACD09] M. Alalfi, J.R. Cordy, , and T.R. Dean. WAFA: Fine-grained dynamic analysis of web applications. In *Proceeding of WSE'2009*, pages 41–50. IEEE CS, 2009.

[CH08] A. Cleve and JL. Hainaut. Dynamic analysis of SQL statements for data-intensive applications reverse engineering. In *Proceeding of WCRE*, pages 192–196. IEEE CS, 2008.

[CMH11] A. Cleve, JR. Meurisse, and JL Hainaut. Database semantics recovery through analysis of dynamic SQL statements. *Journal on Data Semantics*, 15:130–157, 2011.

[CNH12] A. Cleve, N. Noughi, and JL. Hainaut. Dynamic program analysis for database reverse engineering. In *Generative and Transformational Techniques in Software Engineering*, 2012.

[CZvDvR09] B. Cornelissen, A. Zaidman, A. van Deursen, and B. van Rompaey. Trace visualization for program comprehension: A controlled experiment. In *Program Comprehension, IEEE International Conference on Program Comprehension '09.*, pages 100–109, May 2009.

[DG03] M. Debusmann and K. Geihs. Efficient and transparent instrumentation of application components using an aspect-oriented approach. In *Proceeding of DSOM'2003*, volume 2867 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2003.

[JSB] D. F. Jerding, J. T. Stasko, and T. Ball. Visualizing interactions in program executions. In *Software Engineering, 1997*, pages 360–370.

[LD02] M. Lanza and S. Ducasse. Understanding software evolution using a combination of software visualization and software metrics. In *In Proceeding of LMO 2002 (Langages et Modèles Objets*, pages 135–149. Lavoisier, 2002.

[NMMC14] N. Noughi, M. Mori, L. Meurice, and A. Cleve. Understanding the database manipulation behavior of programs. In *Proceeding of IEEE International Conference on Program Comprehension*, page 4, 2014.

[YPZ09] Y. Yang, X. Peng, and W. Zhao. Domain feature model recovery from multiple applications using data access semantics and formal concept analysis. In *Proceeding of Working Conf. Reverse Engineering'2009*, pages 215–224. IEEE CS, 2009.