DAHLIA: A Visual Analyzer of Database Schema Evolution

Loup Meurice and Anthony Cleve
PReCISE Research Center, University of Namur, Belgium
{loup.meurice,anthony.cleve}@unamur.be

***Abstract*—In a continuously changing environment, software evolution becomes an unavoidable activity. The mining software repositories (MSR) field studies the valuable data available in software repositories such as source code version-control systems, issue/bug-tracking systems, or communication archives. In recent years, many researchers have used MSR techniques as a way to support software understanding and evolution. While many software systems are data-intensive, i.e., their central artefact is a database, little attention has been devoted to the analysis of this important system component in the context of software evolution. The goal of our work is to reduce this gap by considering the database evolution history as an additional information source to aid software evolution. We present DAHLIA (DAtabase ScHema EvoLutIon Analysis), a visual analyzer of database schema evolution. Our tool mines the database schema evolution history from the software repository and allows its interactive, visual analysis. We describe DAHLIA and present our novel approach supporting data-intensive software evolution.**

I. INTRODUCTION

In an ever-changing environment, software system evolution is ubiquitous. Over last decades, the research community has largely studied this activity. In particular, it has been shown that understanding the evolution history of complex software systems can significantly aid reengineering and evolution processes. Mining software repositories (MSR) techniques provide an ideal way to extract historical data allowing an in-depth analysis of system evolution. Most research works in the MSR field focus on common software artefacts (e.g., the program source code, bug reports, mail archives, etc.). In contrast, fewer works have considered the database component as main artefact of interest. However, the database occupies an increasingly important place in today's software systems, particularly in data-intensive systems. In this work, we consider that the understanding of the database schema, and of its evolution, may also contribute to the understanding of the system as a whole.

Our main objective in this paper is to recover a precise knowledge of the evolution history of the database schema because it constitutes an important element for gaining an understanding of the database. We propose a fully generic tool-supported approach (1) allowing one to extract such a historical knowledge from a software project repository and (2) proposing an interactive visualization for analyzing database schema history. This approach is implemented through DAHLIA, a visual analyzer of database schema evolution.

II. RELATED WORK

Existing works in this domain analyze the evolution of rather small database schemas. Curino *et. al* [2] present an empirical study of database schema evolution on Wikipedia. They analyze basic information of schema evolution, such as schema size growth, and lifetime of tables and columns. They also provide a classification of schema changes and they study the frequency distribution of those schema changes. The authors propose, in addition to a schema evolution statistics extractor, a tool that operates on the differences between subsequent schema versions and semi-automatically extracts the set of possible schema changes that have been applied. Qiu *et. al* [5] conduct a large-scale empirical study on ten popular database applications to analyse how schemas and application code co-evolve. In particular, they study the evolution histories from the respective repositories to understand whether database schemas evolve frequently and significantly, how schemas evolve and impact the application code.

III. APPROACH

The objective of our tool-supported approach is three-fold: extracting, representing and exploiting the history of a database schema. Our approach consists in extracting and comparing the successive versions of the database schema in order to produce the so-called *historical schema*. This historical schema is a visual and browsable representation of the database schema evolution over time. The global process followed by our approach can be divided into four steps [4]:

- SQL code extraction: we first extract all the SQL files corresponding to each system version, by exploiting the versioning system (e.g., GIT/SVN repository).
- Schema extraction: we extract the physical schema corresponding to each SQL file.
- Historical schema extraction: we build the corresponding historical schema by comparing the successive physical schemas.
- Visualization & exploitation: the historical schema can then be visualized and queried to obtain meaningful historical information about the database schema evolution.

Figure 1 shows two examples of database schema evolution. The left-hand size of the first example (1) illustrates three successive schema versions. Schema S_1 is the oldest one and schema S_3 is the most recent. We can see that between S_1 and S_2, column A_2 has been deleted, column B_2 has been created

as well as table D and its columns. Moreover the entire table C has been dropped. In S_3, table B has disappeared, table D is unchanged, and table C has re-appeared. The historical schema derived from the first example is depicted at the right-hand side. The historical schema is a global representation of all database schema versions, that contains all objects that have ever existed in the entire schema history. Furthermore, each object of the historical schema is annotated with the following meta-attributes:

- $listOfPresence$: the list of schema version dates where the object is present.

- $listOfDeletion$: the list of schema version dates where the object has been deleted.

Example (2) shows a simpler schema evolution. In S_1, table A has columns A_1 and A_2. In S_2, the two columns are still present but the datatype of A_1 has changed (A'_1). The corresponding historical schema must contain this historical information, and therefore we introduce a new kind of objects called *sub-column*. Each *sub-column* represents a datatype change of the *parent-column*. Moreover, a sub-column is annotated with the meta-attribute $creationDate$ corresponding to the schema version date of the change. Up to now, the historical information contained in the historical schema allows one to identify 13 categories of atomic database changes between successive schema versions: adding/dropping a table; adding/dropping a column; changing the column datatype; adding/dropping an identifier; adding/dropping an index; adding/dropping a foreign key; renaming a column or table.

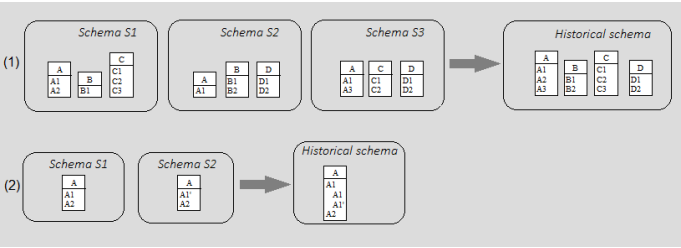

Fig. 1. Two schema evolution examples and their corresponding historical schema.

DAHLIA provides the user with a visual and browsable representation of the database schema evolution history. It takes the historical schema as input and allows one, among others, to (1) compare two arbitrary schema versions, (2) extract the database schema at a given date, (3) extract the complete history of a particular schema object (column/table), (4) extract various statistics about the evolution of the database schema, (5) analyze the involvement of each developer in that evolution. DAHLIA is implemented as a Java plugin of DB-MAIN [3]. DB-MAIN is a generic database engineering tool with integrated support for database design, reverse engineering, re-engineering, integration, maintenance and evolution.

IV. DAHLIA: A VISUAL ANALYZER OF DATABASE SCHEMA EVOLUTION

DAHLIA provides two visualization modes: 2D and 3D. Figure 2 depicts the main 2D user interface, after having loaded a given historical schema. This mode proposes an interactive panel allowing one to manipulate the physical objects of the historical schema and query their respective history. This historical schema includes all physical objects that have ever existed in the entire schema history: tables, columns, identifiers (ID), foreign keys (FK) and indexes (INDEX).

The 3D mode makes use of the well-known city metaphor of CodeCity [6] (see Figure 4 for an example). In DAHLIA, a building represents a table, the number of columns is mapped on the building height while the base size represents the number of schema changes happened during the table life. The mapping between the database schema metrics and the visual metrics can be (re)configured by the user. That 3D visualization is particularly suitable for analyzing very large database schemas.

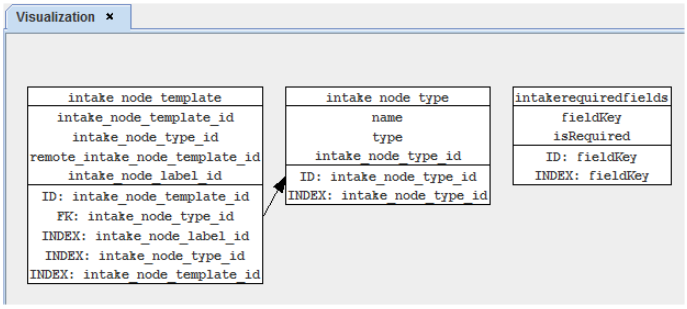

Fig. 2. An example of historical schema - 2D visualization.

A. Filtering and Comparison

DAHLIA proposes the user a filtering functionality allowing displaying the database schema version corresponding to an input date. A *schema-diff* feature is also implemented and gives the opportunity to compare two different schema versions. It helps the user to analyze how the schema has evolved over time.

B. History Analysis

DAHLIA offers diverse features for extracting historical knowledge about schema evolution. The first one gives a first insight about the longevity of the physical objects. Our tool assigns a colour to each object depending on its age and liveness. All objects depicted in green constitute the objects which are still present in the latest schema version, while all red objects have been dropped. The colour shades corresponds to the age of the objects. A dark red object is an object that has been dropped a long time ago, whereas a light red object is an object that has recently been removed from the schema. The darker the green, the older the object is, and vice versa. Figure 3 and 4 show an example of colourized historical schema as it can be viewed, respectively, in 2D and 3D mode.

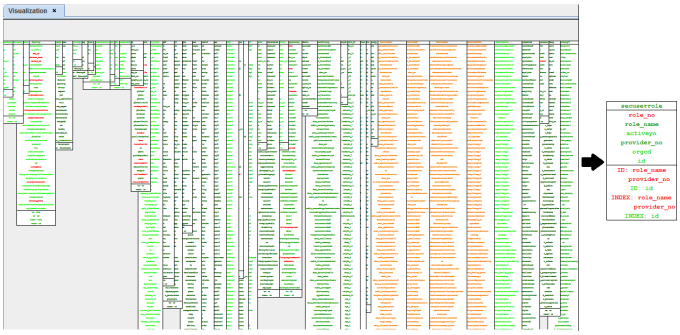

Fig. 3. An example of colourized historical schema displayed in DAHLIA 2D mode. The table *secuserrole* is a particular table of the historical schema.

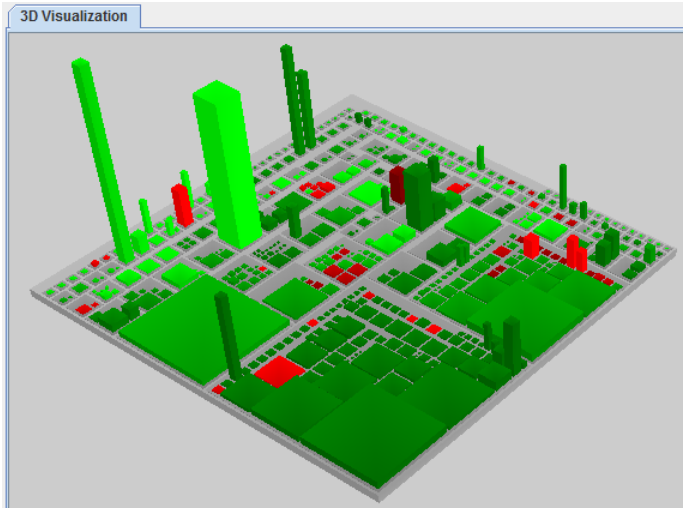

Fig. 4. An example of colourized historical schema displayed in DAHLIA 3D mode.

The second category of historical knowledge that can be extracted is related to the change history of a given object. Indeed, during its lifetime, a schema object may be subject to several successive changes, including creation, deletion, modification, reappearance (re-created after having been dropped). DAHLIA permits to extract and analyze such a change history.

C. Statistics Extraction

Various statistics about the evolution of the database schema can be automatically obtained thanks to DAHLIA. The user can easily analyse the general trends of the schema evolution such as the evolution of the number of the tables/columns over time, the number of created/deleted physical object through the schema versions, etc. Other statistics related to the developers involvement can be visualized with DAHLIA. Those statistics allows the rapid identification of the database schema experts, as well as the developers that are the most familiar with a given schema fragment. The tool also measures the stability of the tables and columns, which can be useful in the context of database migration. A table that has been created a long time ago,and that has not been subject to frequent modifications, can be considered stable. DAHLIA also provides statistics about the different change types applied to the schema during its lifetime.

D. Support for Implicit Renaming Detection

Detecting a table/column renaming is an easy process when the SQL migration scripts between successive schema versions are available. However, in case of absence of such scripts, the task becomes much more complex. Indeed, if table A is renamed as B, there is no direct way to detect it and the historical schema considers that table A has been dropped while table B has been created without keeping a link between both tables. In such a case, we see that a finer-grained approach is required. This is why DAHLIA proposes a semi-automatic approach supporting the identification of implicit (column/table) renamings. Our visualization tool computes the most likely *implicit* renamings according to different comparison criteria (e.g., name similarity, the column type similarity, etc.). This problem can be expressed as an optimization problem subject to constraints solvable with a well-known mathematical method called *linear programming* (LP). LP is a specific case of mathematical programming allowing one to achieve the best outcome in a given mathematical model.

V. Dahlia Applied to Real-Life Case Studies

We selected 4 popular open-source database applications from different domains, and we used DAHLIA to analyze the evolution of their database.

- OSCAR system: OSCAR (Open Source Clinical Application Resource) is full-featured Electronic Medical Record (EMR) software system for primary care clinics. It is widely used in hundreds of clinics across Canada. Figure 3 depicts the historical schema of OSCAR. The OSCAR system is further analyzed in [1].
- MediaWiki: MediaWiki is a free and open source wiki software, used to power wiki websites such as Wikipedia, Wiktionary and Commons, developed by the Wikimedia Foundation and others.
- TikiWiki: TikiWiki is a free and open source wiki-based, content management system and Online office suite.
- PrestaShop: PrestaShop is a free, open source e-commerce solution.

Table I describes the 4 studied projects, the study period and their evolution. The history of OSCAR and MediaWiki has been extracted from their respective GitHub repository, while PrestaShop and TikiWiki use SVN as version control system. DAHLIA allows the automatic extraction of the schema history from those two popular version control systems, i.e., Git and SVN. Table II shows, for each project, the distribution of the 13 atomic database change types defined in Section III. The most frequent operation is the modification of the column datatype. Furthermore, one notices the general evolution trend is adding new structures, especially columns, tables and indexes. The developers rarely remove existing data structures. The figures of Table II have been automatically produced by DAHLIA.

TABLE I
The 4 studied database applications and their evolution.

Project	Studied Period	#Tables	#Columns	#Versions
OSCAR	07/2003 → 06/2013	88 → 445	2443 → 13364	670
MediaWiki	05/2003 → 08/2013	17 → 50	100 → 337	359
TikiWiki	12/2006 → 07/2013	206 → 248	1525 → 1974	623
PrestaShop	12/2008 → 09/2012	113 → 157	564 → 890	229

TABLE II
Distribution of schema changes.

Change type (%)	Oscar	MediaWiki	TikiWiki	PrestaShop
Adding table	9.7	9.6	19.4	19.6
Dropping table	1.5	3.9	3.4	1.7
Adding column	28.7	15.7	14.7	14.9
Dropping column	3.5	5.4	2.5	2.6
Adding ID	0.8	2.5	1.6	2.7
Dropping ID	0.3	0.9	1.4	0.7
Adding FK	0.05	0	0	0
Dropping FK	0.2	0	0	0
Adding index	2.3	12.5	5.4	14.7
Dropping index	0.4	4.3	2.3	2.3
Changing column datatype	41.6	44.1	48.8	39.6
Renaming table	0.2	0.11	0.1	0.1
Renaming column	10.6	0.9	2.5	1

As previously explained, DAHLIA allows an in-depth analysis of the evolution of a given schema object. Figure 5 depicts a particular fragment of the OSCAR historical schema. It shows the complete history of column $PROGRAM\_ID$. The column has been created on 29 April 2012 by the developer identified by $matthew.ma20110628@gmail.com$. Then, the column has been removed on 15 February 2013 by $anniezhou91@gmail.com$ and has immediatly been re-created on the same day by $marc@mdumontier.com$. The column has been finally dropped on 25 February 2013 by the same developer. This evolution history gives additional information about how the object has evolved over time. But more generally DAHLIA helps answering some primordial evolution questions such as "who are the specialists of that table/part of the database?", "who is the most appropriate (e.g., most experienced or most qualified) person for achieving a particular database-related activity?".

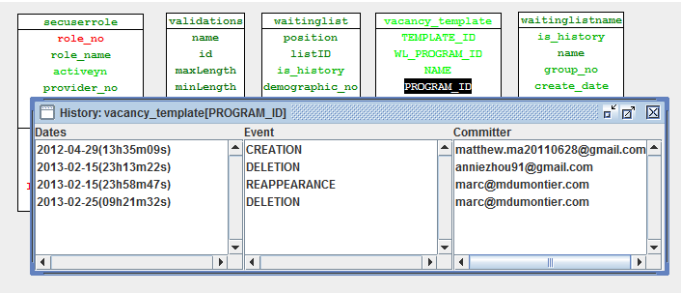

Fig. 5. The complete history of column $PROGRAM\_ID$

Figure 6 shows the involving of the developers of each studied project. Each slice corresponds to a developer and its size represents the number of distinct tables impacted by the developer (by creating, updating or deleting the table). Those figures have been automatically generated by DAHLIA. One can see the distribution of the OSCAR and MediaWiki developers is quite homogeneous (fewer specialists), whereas the majority of the database changes of TikiWiki and PrestaShop is performed by a single person.

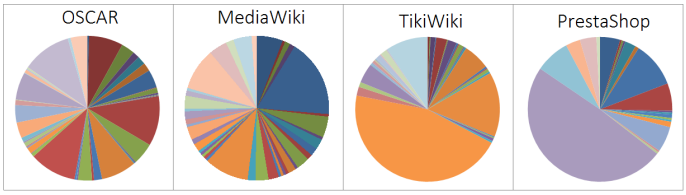

Fig. 6. Information about the developers involved in the evolution of the database schema - a slice corresponds to the number of tables impacted by a developer (one slice per developer).

VI. Conclusion

We presented DAHLIA, a novel visualization tool that allows us to analyze the evolution history of a database over its lifetime. This work definitely opens several important new research and collaboration perspectives for the entire software evolution research community. Indeed, considering the link between the evolution of the database and the evolution of all the other software artefacts remains a largely unexplored yet important research domain. Thanks to DAHLIA, we could conduct empirical studies on other projects in order to answer some general evolution questions such as: (1) How do database schemas evolve? (2) What are the most common evolution patterns that can be observed within and across data-intensive software systems (DISS)? (3) How do database (schemas) and programs co-evolve in a DISS? (4) What are the most frequent co-evolution patterns that can be observed?

Acknowledgements.

This work has been supported by the F.R.S.-FNRS, in the context of the DISSE research project, carried out in collaboration with the University of Mons.

References

[1] A. Cleve, M. Gobert, L. Meurice, J. Maes, and J. Weber. Understanding database schema evolution: A case study. *Science of Computer Programming*, 2014.

[2] C. Curino, H. J. Moon, L. Tanca, and C. Zaniolo. Schema evolution in wikipedia - toward a web information system benchmark. In *ICEIS (1)*, pages 323–332, 2008.

[3] DB-MAIN. Official website. http://www.db-main.be.

[4] M. Gobert, J. Maes, A. Cleve, and J. Weber. Understanding schema evolution as a basis for database reengineering. In *Proceeding of 29th IEEE International Conference on Software Maintenance (ICSM 2013)*. IEEE CS, 2013.

[5] D. Qiu, B. Li, and Z. Su. An empirical analysis of the co-evolution of schema and code in database applications. In *Proc. of ESEC/FSE*, pages 125–135, New York, NY, USA, 2013. ACM.

[6] R. Wettel and M. Lanza. Codecity: 3d visualization of large-scale software. In W. Schfer, M. B. Dwyer, and V. Gruhn, editors, *ICSE Companion*, pages 921–922. ACM, 2008.

APPENDIX

During the tool demonstration, we will present DAHLIA and its contribution to support the analysis of data-intensive software evolution.

1) We will first observe the MediaWiki database evolution by showing all the physical schemas (and their differences) resulting from the schema extraction phase, described in Section III). Those schemas will have been extracted beforehand from the project's repository.
2) Then, we will build the corresponding historical schema (historical schema extraction phase of Section III). During that extraction phase (about 2 minutes), we will explain how the process automatically builds the historical schema.
3) We will finally use DAHLIA to exploit and visualize the historical schemas of the 4 case studies (generated beforehand, due to timing reasons) presented in Section V. We will study the schema evolution of those 4 case studies. We will make a time travel thanks to a particular feature allowing us to successively display all the schema versions of the studied project (Figure 7). We will also show how DAHLIA enables to study the differences between two selected schema versions (Figure 8). We will then prove how the tool helps the user to recover some implicit renamings, requiring user inspection and validation. Figure 9 shows the list of implicit table renamings proposed by DAHLIA for the OSCAR project. In particular, one can see our tool has detected that there is a potential renaming occurred at schema version 471: table *specshis* is renamed as *eyeformspecshistory* and the underscore characters appearing in the column names have been removed.
   In conclusion, during the demonstration, we will demonstrate how DAHLIA may help the user to answer several evolution questions such as how does the database schema evolve over time? which is the general evolution trend? which are the most stable tables? who are the most qualified developers for achieving a given schema evolution phase?.

Tool availability: DAHLIA is available for download at http://info.fundp.ac.be/~lmeurice/DAHLIA. Installation and usage instructions are provided at the same address.

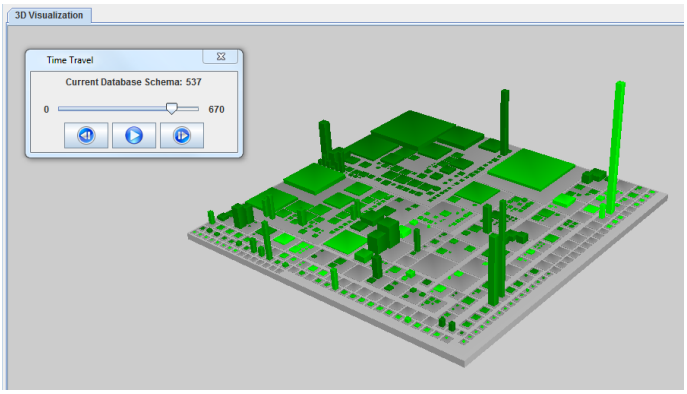

Fig. 7. Time travel - Successive display of the different schema versions.

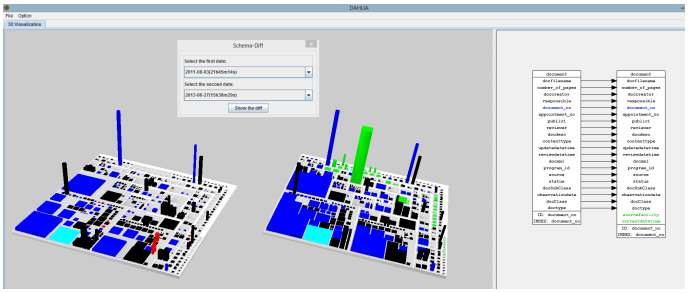

Fig. 8. Example of schema-diff - 2 OSCAR schema versions (2011-08-03 and 2013-06-27).

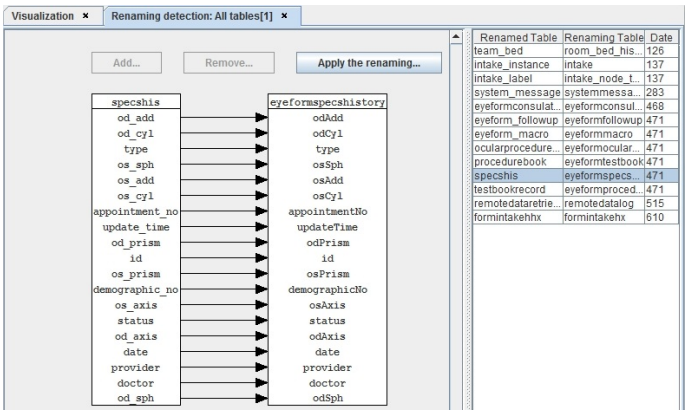

Fig. 9. The implicit table renamings automatically detected by DAHLIA.