Digging Deep: Software Reengineering Supported by Database Reverse
Engineering of a System with 30+ Years of Legacy

Stefan Strobl, Mario Bernhart, Thomas Grechenig
Research Group for Industrial Software
Vienna University of Technology
{stefan.strobl, mario.bernhart, thomas.grechenig}@inso.tuwien.ac.at

Wolfgang Kleinert
Information Technology Services
Vienna University of Technology
wolfgang.kleinert@tuwien.ac.at

Abstract

This paper describes the industrial experience in performing database reverse engineering on a large scale software reengineering project. The project in question deals with a highly heterogeneous in-house information system (IS) that has grown and evolved in numerous steps over the past three decades. This IS consists of a large number of loosely coupled single purpose systems with a database driven COBOL application at the centre, which has been adopted and enhanced to expose some functionality over the web. The software reengineering effort that provides the context for this paper deals with unifying these components and completely migrating the IS to an up-to-date and homogeneous platform.

A database reverse engineering (DRE) process was tailored to suit the project environment consisting of almost 350 tables and 5600 columns. It aims at providing the developers of the software reengineering project with the necessary information about the more than thirty year old legacy databases to successfully perform the data migration. The application of the DRE process resulted in the development of a high-level categorization of the data model, a wiki based redocumentation structure and the essential data-access statistics.

1. Introduction

The legacy system in question was a mid sized in-house information system (IS) with the oldest parts dating back to the late sixties. Since then numerous developers have made countless changes to it, many of them undocumented rendering the original (and only) documentation useless. In recent years it became apparent, that the current system was increasingly difficult and costly to maintain and enhance. Therefore the decision was made to reengineer and unify a large part of this system. The heart of this environment was a large database system consisting of two main and several additional smaller databases. These main databases were originally designed as COBOL VSAM files and then later migrated to a commercial relational database management system (RDBMS). While this was a step in the right direction, the existing structure and quality of the database remained unchanged as the following section (3) will demonstrate.

Due to the fact that the database migration was rather neglected during the early stages of this software reengineering project and due to the usual tight project schedule it was unavoidable to perform the database reverse engineering in parallel with the development of the new system. For this supporting database reverse engineering effort the following goals were set:

- **Design** a database reverse engineering process suitable for supporting a large scale ongoing software reengineering project.

- **Document** the process in a generic and reusable manner.

- **Apply** the process in the context of the software reengineering project.

- **Evaluate** the effect the process had on the project and its outcome.

After summarizing the related work in this field of research, the remaining paper will be organized as follows: Section 3 will give a detailed view of the problem this paper deals with. Subsequently section 4 will present a five step process that attempts to solve it. The paper is then summarized by a short section qualitatively evaluating the effects of the chosen approach.
2. Related Work

Davis and Aiken [5] have published a very comprehensive history of the field of data reverse engineering. Müller et al. have published a roadmap [8] for the field of reverse engineering for the first decade of the 21st century pointing out that apart from perfecting reverse engineering tools and methodologies the most important task will be to teach reverse engineering. The process of reverse engineering databases depicted in this paper has been influenced by the earlier work of Blaha [2] and Henrard [7].

3. Problem Definition

The legacy databases that have to be reverse engineered have a wide range of deficits making analysis and migration of the data structures and their contents difficult. The following set of issues has been identified:

3.1. Outdated or Missing Documentation

One of the most common issues in reverse engineering as already identified by Blaha [3] is poor documentation. In this case the legacy system was scarcely documented to begin with and what little documentation had been created when the system was first developed was not maintained. Therefore the only viable sources of documentation were the remaining developers of the legacy system along with the legacy systems structure and code itself.

3.2. Missing Normalization

Due to the age of the database structure and the fact that it had been originally designed for a file-based database system, the rules of database normalization were not enforced. In fact the database doesn’t even satisfy the First Normal Form [4] and doesn’t have a single defined foreign key constraint.

3.3. Data Model Deprecation

Many elements of the legacy database are not being used any more. Combined with the previously mentioned lack of documentation and in the absence of identifiable naming conventions this issue introduces significant uncertainties. In addition, it causes a substantial overhead as the amount of database assets that have to be analysed is increased unnecessarily.

3.4. Data Application Responsibility

Much of the data used by this system is also used by other systems, often in a separate database. Therefore it is frequently unclear, which system is responsible for keeping the data in a globally consistent state. This makes it much harder to elaborate a reliable and accurate migration path.

4. Solution

The following process mainly based on [2] and [7] has been tailored to achieve the goals defined above. It is structured into five parts, four of which are designed to be executed sequentially. None the less care has been taken to allow an iterative revisiting of each process step at later times. The fifth part, presentation of results, is actually a cross section that is executed in parallel.

4.1. Current State Analysis

Initially neither the extent nor the complexity of the legacy databases was known. Therefore it was necessary to create a comprehensive overview of the current structure. This step of the process was supported by off the shelf CASE (Computer Aided Software Engineering) tools. These tools were of great help for extracting the meta-data (or source physical schema in [7]) describing the database structure. In the end the structural meta-data of 346 tables with a total of around 5600 columns was extracted. Furthermore the comments stored directly in the database system were also transferred to a human readable and distributable format.

4.2. Database Object Categorization

The next step was to fully categorize the database assets discovered by the previous analysis. Therefore three main categories were defined:

- Relevant business data: all data that has to be migrated to the new system as otherwise a loss of business value would be unavoidable.
- System and implementation specific data: all data that is only needed for the legacy system to operate properly, but will not be migrated.
- Deprecated database assets: all aspects of the legacy databases that do not represent any semantical or operational value to either the legacy or the reengineered system.

This categorization is absolutely crucial to the success of the remaining steps and therefore the successful execution of the reverse engineering process as a whole. Special care has to be taken to identify as many deprecated database assets as possible, but at the same time avoid false positives (marking still needed assets as deprecated) at almost all costs.
The distinction between operational data and deprecated items is relevant, as the operational aspects of the database still have to be regarded during the reengineering phase. It might not be possible to fully understand existing functionality, which has to be reengineered to the new system, without understanding the respective database areas.

The significance of this step is relativised to some extent by the fact that this step is also the reentry point for later iterative refinements. The subsequent two steps will certainly unveil detailed information that will help to further refine the categorization produced by the first run of this step.

### 4.3. Data Reverse Engineering

Data Reverse Engineering (DRE), a terminology coined by Aiken in [1], describes the focus of a reverse engineering effort on data assets. This part of the reverse engineering process therefore focuses on the (semantical) understanding of the data to be migrated to the reengineered system. Unlike the first two steps which mainly aim at providing a comprehensive overview, this one focuses on the details. This step provides the engineer migrating the legacy database with the knowledge to do so without altering the semantics of the data.

By (potentially) going into great detail, this step can easily turn out to be massively labour intensive. Therefore finding and evaluating suitable CASE tools to reduce the amount of manual labour should have a high priority. Today a wide variety of open source and commercial data profiling tools are available. None the less a good deal of manual interaction is necessary. Especially the elicitation of implicit foreign keys [6] and the detection of redundancies across database tables still requires manual assessment.

The result of this step is a comprehensive and detailed documentation of the legacy database. With regard to the categorization performed in the previous step the business data has the highest priority and should get the deepest analysis.

### 4.4. Usage and Traffic Analysis

This last step of the reverse engineering process is also the most vaguely defined one. For the project at hand, two factors mostly influenced its characteristics. First the elicitation of deprecated database tables in the categorization step had yielded ambiguous results in several cases, mostly introducing false positives. By recording access to all database tables over a long enough period of time, false positives were largely eliminated and a significant indicator towards false negatives was produced.

In addition the performance test team had severe issues estimating the amount of traffic for certain areas of the legacy application due to the heterogeneity of the methods of accessing the data. Therefore a secondary goal of this step was to provide rough estimates of transaction volumes on several business domains of the reengineered system. Utilizing this data allowed the test team to set realistic performance goals early in the planning stages.

To achieve the goals described above, the production database was configured to collect the appropriate data and produce a daily report. The data gathered in this way was then collected over more than six months and evaluated on a regular basis.

### 4.5. Presentation of Results

As mentioned earlier unlike the other steps of the process, this task can be executed during or after each step. Why was this parallelism introduced instead of including this task as part of each of the other steps? This separation allows for a flexible “just-in-time” delivery of artifacts for concrete purposes. While the weekly project status report might just require a table summarizing the results of the reverse engineering effort, a high level presentation might require a sophisticated graphical representation of the same results.

During the application of this process the following artifacts were produced:

One of the most valuable outputs of the process were the results of the current state analysis (section 4.1) and the categorization effort (section 4.2). The first step yielded a graphical representation of all tables of the legacy databases, although in no particular order. However after performing the categorisation step it was possible to include the results in the diagram from the previous step, increasing the expressiveness of the presentation significantly. By simply arranging the tables by category and at the same time colour coding each of the categories, a universal diagram was created that visualized some of the key aspects of the migration effort. It demonstrated to none-technical stakeholders the fact, that more than 40 percent of the legacy database assets would not be included in the migration.

A second and equally useful output was the web based redocumentation of the legacy database. On the basis of the results of the first two steps in the process a skeleton structure was generated in a wiki. This skeleton consisted of one page for each table in the database as well as a series of indexes for easy access. The indexes were built representing the categorization. Each page describing a single table was already generated to include all the information available from the previous steps. During the execution of the data reverse engineering step (section 4.3) the analysts then entered the newly retrieved information into these pages. After completing the third step of the process, the documentation was a fairly complete and up to date source of infor-
Figure 1. The combined graph for presenting the results of the analysis - generated for each table.

5. Evaluation

To assess the effect of the database reverse engineering process on the overall project performance, a series of interviews was done with different groups of stakeholders. These interviews showed that different stakeholder groups benefited from different steps in the process. The higher level, decision making stakeholders were largely interested in the results of the first two steps. These steps provided them with relatively concrete numbers so they were able to better estimate the remaining efforts.

The development and migration team, which incidentally also performed the data reverse engineering step, benefited the most from the effort. This step yielded exactly the kind of information they needed to write the code for migrating the data from the old to the new database structure. In addition the web-based documentation was well equipped to handle an incremental and exploratory style of performing the data profiling. This way, many developers were able to contribute information about a single database table.

The third group of stakeholders benefiting from the results of the process at a regular basis were the requirement engineers. Although they were not usually involved at such a technical level, the documentation was frequently consulted to answer data centric questions. Furthermore they provided the developers with inputs as to where the documentation was still incomplete or not precise enough.

6. Conclusions

Summarizing the design and implementation of a new process for database reverse engineering in a supporting role to a larger software reengineering project has yielded the following results. First and probably most importantly the legacy databases were fully captured and documented. All the captured assets were then categorized to elicit deprecated aspects and to structure the remaining ones by business domain. A comprehensive and detailed web-based re-documentation effort was launched to fully capture all relevant aspects of the legacy databases. A preliminary usage analysis was conducted to verify the categorization and documentation effort and to further improve the accuracy of the information gathered during the earlier steps of the process. Finally the interview based evaluation demonstrated how the stakeholders of the project benefited from the effort.

References