A Validatable Legacy Database Migration Using ORM

Tjeerd H. Moes[1], Jan Pieter Wijbenga[1], Herman Balsters[2], and George B. Huitema[2]

[1] TNO
P.O. Box 1416 9701 BK Groningen,
The Netherlands
{tjeerd.moes,jpwijbenga}@tno.nl
[2] University of Groningen
Faculty of Economics and Business
P.O. Box 800 9700 AV Groningen,
The Netherlands
{h.balsters,g.b.huitema}@rug.nl

Abstract. This paper describes a method used in a real-life case of a legacy database migration. The difficulty of the case lies in the fact that the legacy application to be replaced has to remain fully available during the migration process while at the same time data from the old system is to be integrated within the new system. The target database schema was fixed beforehand, hence complicating and limiting our choices in constructing a possible target schema. The conceptual approach of the Object-Role Modeling (ORM) method helped us to better understand the semantics of the source and target system and enabled us to abstract from implementation choices in both the source and the target schemas. We discuss how our method could help in executing other legacy data migration projects.

Keywords: data migration, conceptual modeling, fact-oriented modeling, re-engineering, ORM, database schema mapping.

1 Introduction

Legacy IT systems form a growing problem for companies and other institutions. Their core business usually depends on legacy systems while at the same time knowledge of these systems may be fading or absent. Furthermore, as the organization changes, the legacy system may fall behind in terms of functionality, adaptability, reliability or user interface. Businesses must make an inconvenient tradeoff between the growing risk of using legacy systems and the often bigger risk of migration to newer systems. Considering the prevalence of legacy systems, one can conclude that many companies choose to stick with their legacy systems.

When, for whatever reason, a migration becomes inevitable it is critical that the migration process is well-manageable as well as well-managed. This is why legacy migration has received considerable attention in the literature, see [5], [12], [13] and [15].

P. Herrero et al. (Eds.): OTM 2012 Workshops, LNCS 7567, pp. 409–418, 2012.

The problem of database re-engineering stems from (legacy) databases that are hard to understand or that perform badly. Database re-engineering involves the reconstruction of the (intended) semantics of some source database. Subsequent data migration involves mapping a source database schema to a target database schema. We shall show how Fact-based modeling, and in particular ORM, can help in re-engineering (relational) databases.

Fact-based modeling (FBM [6]) is a methodology for modeling the universe of discourse (UoD) of an information system. The main purpose of fact-based modeling is to capture the semantics of the UoD, and to validate results with a domain expert. Unlike Entity-Relationship (ER) modeling or object-oriented modeling, fact-based modeling treats all facts as relationships (unary, binary, ternary etc.). Fact-based modeling facilitates natural verbalization and thus enables productive communication (and, hence, validation) between stakeholders. FBM is also used as the general name of several fact-based conceptual data modeling dialects, such as Object Role Modeling (ORM [9]), Natural language Information Analysis Method (NIAM) [8], and Fully-Communication Oriented Information Modeling (FCO-IM) [1]. This paper uses basic ORM notations; the version of ORM discussed in this paper is ORM 2 [7], as supported by the NORMA tool [3], [4].

This paper provides a first case of applying a migration method using fact-oriented modeling for legacy databases after a method by [14] (explained in Section 3). Moreover our paper proposes a number of improvements to this method. It is structured as follows. Section 2 introduces the migration case. The developed migration method is shortly described in Section 3. Section 4 states the results and the experiences of applying this method. Finally, in Section 5 the conclusion is presented together with some guidelines for future research.

2 Case Description; Migration of a Legacy Online Survey Tool

A sales company, here referred to as SalesCo, uses an online survey tool to gather information on customer satisfaction. This monitoring tool, called KBM, produces on average 1000 survey invitations and processes 150 customer responses per week. The response data is used by SalesCo to manage the customer service and sales departments.

The tool KBM, developed in 2000, must be replaced by a newer and better survey tool LimeSurvey (LS). Continuing to operating the legacy KBM system imposes a big risk since the KBM system administrator is the only person who knows how to operate and maintain KBM. Moreover no system documentation exists at all.

Because of its importance to SalesCo and the frequent use by its customers, the service that KBM provides and the data it contains, must remain operational and accessible at all times. Therefore we have to find a migration method that will eventually result in a complete shutdown of KBM while SalesCo can continue to do online surveys and access the response data on the new LS system.

3 The Applied Migration Method; Re-engineering Steps

The migration method we apply in this paper is based on the experimental method by [14]. In this method the migration is performed on the conceptual level before the

physical migration is started. We use elements of that method, add detail to the existing steps and add an extra step on the conceptual level that creates a conceptual model by following the conceptual schema design procedure CSDP [9].

In this section we will describe 5 steps that we conducted to better understand the source and target schema and design an initial mapping between the two. We call this the re-engineering phase of the migration. The steps of re-engineering the source and target schemas and creating the required mappings are as follows. Figure 2 visualizes the re-engineering phase and the created artifacts, illustrated by examples from the case. The prefix 'kbm_' stands for the source system; 'ls_' indicates the target system. A prefix of 'm_' indicates that the object type originates from the optimized model.

Step 1: Creating an Optimized and Correct Conceptual Model of Data to Be Migrated (Artifact A1). We select the subset that only contains data that has to be migrated from the source database. This model is created as if there is no target database to take into account and it abstracts from specific implementation choices of the source. Hence, this is the ideal target model and it is created using the CSDP [9] and input from the business analysis phase. The CSDP ensures that we capture the semantics in a complete and correct fashion. In this model we ignore parts of the source schema that are non-normalized.

Step 2: Creating an Unoptimized but Valid ORM Model of the Source Database (Artifact A2). The first purpose of this step is to help understand the semantics of the source schema and provide useful knowledge to extract the data correctly. This step uses an unoptimized ORM model (Artifact A2) that has correct ORM syntax. It is called unoptimized because it is not constructed using the CSDP, but using a simple reverse engineering procedure that has two substeps. First, we iteratively construct an ORM model where tables in the source schema are represented by object types that have binary relations with value types representing the table's columns. Secondly, we identify and add to the model all constraints that were left implicit in the original source database schema. These missing constraints may or may not be enforced in the application logic. Knowing these constraints simplifies data extraction. The second purpose of this step is that it indicates problems regarding data pollution and the presence of quirks in the schema. This provides valuable input for the process of cleaning the data and the source schema. For brevity reasons, we will not address the cleaning of data here.

We use an ORM-based questionnaire [14] to provide a systematic way to identify which tables and columns should be migrated from the source to the target database, as illustrated in Figure 1.

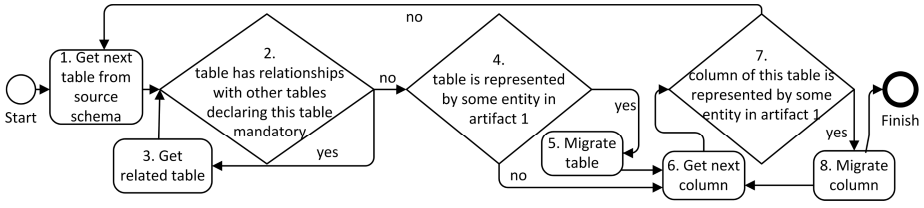

Fig. 1. ORM based questionnaire, after [14]

Artifact A1 (the ORM conceptual model of data to be migrated) serves as the first input for the procedure. It answers the question; which tables must be migrated? The second input for the procedure is the source database schema. It provides all available tables and columns.

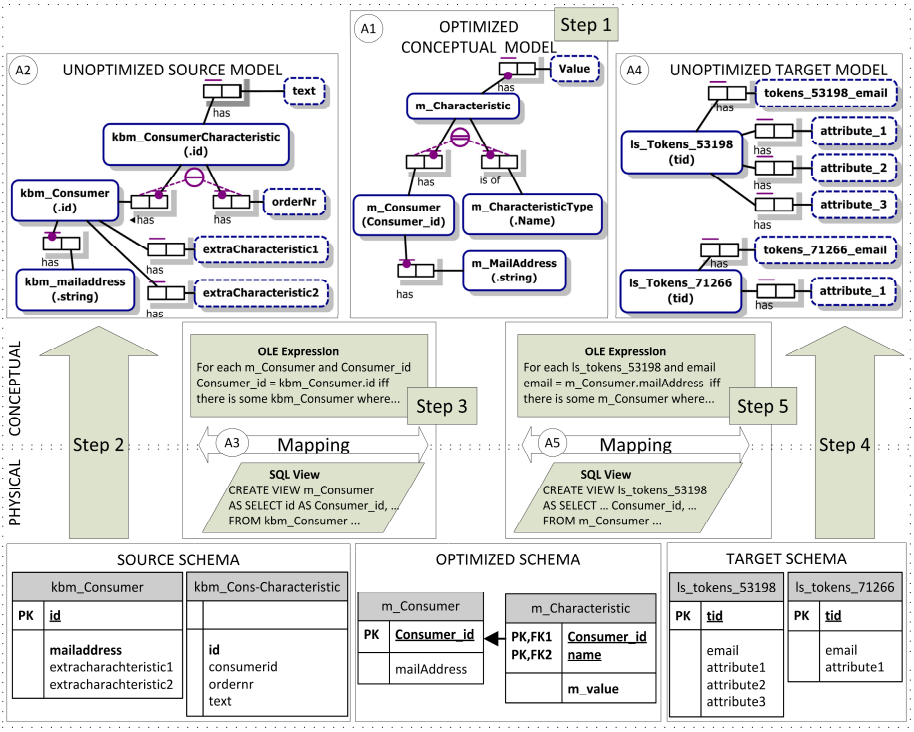

Fig. 2. The re-engineering and mapping phase of the migration method with process steps and artifacts

Step 3: Creating a Mapping between the Optimized Conceptual Model's Schema and the Unoptimized Source Schema (Artifact A3). The optimized and correct conceptual ORM model (A1) is mapped to the unoptimized ORM model (A2). This mapping uses sentences written in a structured format, coined as OLE: ORM Logic-based English [2] (not to be confused with Object Linking and Embedding). OLE aims at being easy to read and write by non-technical domain experts (users of the database), hence facilitating validation purposes. OLE is used to specify, in structured natural language format, elementary facts and constraints in ORM. For migrating from source to target models, we specify each fact type in a target table in terms of fact types coming from the source model. These specifications are entered in OLE

into an experimental tool called the ORM ReDesigner that will automatically transform these OLE specifications of (derived) fact types into view definitions in SQL.

Step 4: Creating an Unoptimized ORM Model of the Target Database (Artifact A4). Similar to step 2, the resulting model provides valuable knowledge about the target schema for creating the mapping.

Step 5: Mapping between the Optimized Conceptual ORM Model's Schema and the Unoptimized ORM Target Schema (Artifact A5). OLE expressions map the optimized and correct conceptual ORM model (A1) to the unoptimized ORM target model (A4).This mapping facilitates insertion of the data into the target system.

After the re-engineering phase, the physical migration phase is started. A snapshot of the source schema is copied into a temporary migration database running on the target database server. A synchronization mechanism between source and temporary migration DBMS is implemented to propagate transactions between the two servers. The mapping A3 is implemented in the temporary migration database and the mapping A5 is implemented in the target database. A transaction propagation mechanism provides the final integration of data from the mapping A5 to into the target database.

4 Application of the Migration Method

KBM Database Details. The systems KBM and LS are running on different physical machines. Both systems use the same database management system (SQL Server 2008) for persistent data storage. The present system administrator explained that during the 12 years of operation KBM has received several ad-hoc database and application updates to meet new survey requirements.

All survey data, together with application data, is stored in the KBM database that contains 40 tables. An overview of the database schema shows some remarkable implementation choices: No primary key is defined for ten of the tables, 18 tables lack foreign key constraints that should be there or are improperly not referenced by a foreign key constraint in another table. Also, data relating to the same conceptual object type is spread over multiple tables.

Figure 3 below illustrates these findings. Tables 'optout' (short for 'optional out') and 'consumercharacteristic' do not have a primary key and both tables do not have any explicit relation to other tables. For the table 'consumercharacteristic' we can guess, based on the used naming convention, that the attribute 'consumerid' references the id in the table 'consumer' as a foreign key constraint. Probably, the mail addresses in 'optout' also relate to consumer, but we will come back to this later. The table 'providedanswer' stores response data of all surveys and does not contain a primary key or foreign key constraint either.

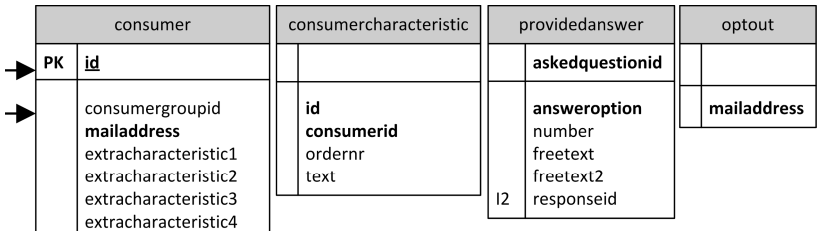

Fig. 3. Part of the ER model of the KBM database schema, (the arrows indicate a foreign key reference from another table not shown in this fragment, 'PK' indicates primary key)

The non-mandatory attributes 'extracharacteristic1' up to 'extracharacteristic4' are part of the 'consumer' table. But it appears that a consumer can have any number of characteristics since the 'consumercharacteristic' table also stores characteristics of consumers. So, these characteristics are stored over two tables, although there is no semantic difference.

LS Database Details. The target system LS is developed by the open source community of LimeSurvey.org. The LS database contains 24 base tables that store survey and application data. For each new survey two tables are added –called 'tokens_...' and 'survey_...' - to the database. These tables contain customer invitation data and survey response data. When we compare the database by the same implementation aspects as we did for KBM we get the following results: A primary key is defined for all 24 tables; There are no foreign key constraints in the schema and attributes relating to the same entity are properly located in the same table, as it seems for now.

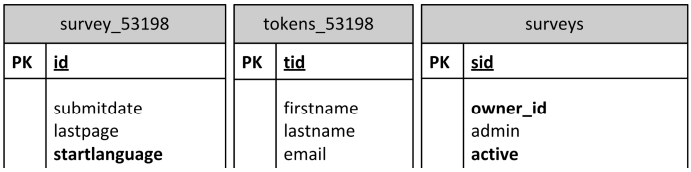

Fig. 4. Part of the ER model of the LS database schema, only the first 7 attributes of each table are shown

Based on the table names we can guess that the last part (53198) of the 'survey_53198' and 'tokens_53198' table's names is a foreign key in the 'surveys' table.

This short analysis took the project team data model expert 1 day to make. Based on this analysis we conclude that many constraints on data of both databases are stored in application logic. We have to work with the system as is, and take the findings into account during the project execution.

4.1 Re-engineering Phase

Deriving semantics from the KBM database was difficult. No documentation was available and the system administrator could not provide detailed information about the meaning of tables and attributes either. So, understanding the semantics of the plain schema of the source database –containing 265 attributes in 40 tables- proved to be very hard. Looking at the source code of the application did not bring the team to a better understanding either. The application code contained many 'hacks' and was not systematically written using some particular design paradigm.

Analyzing the target LS system, the team found out that LS utilizes a completely different schema compared to that of KBM. Although LS was well-documented and the database schema was easier to understand than KBM's, it was hard to point out the relation between attributes in the target database schema and the attributes in the source schema. Apparently, the two databases were implemented very differently.

Re-engineering Step 1. To cope with different schema implementations, the team constructed an optimized and correct conceptual model that represents the information to be migrated in a fashion that is independent of implementation.

After 12 iterations and a final validation with the domain expert, the model contained fifteen entity types, eight value types and fifty roles. This model was Artifact A1 (see Figure 1); it contained the subset of all data required in the target system. Mapping this conceptual model to a relational schema (using the RMap procedure) provided 70 columns in 13 tables. This schema, together with semantic information from the ORM conceptual model, was validated by SalesCo and the project manager. Figure 5 shows a part of the first artifact.

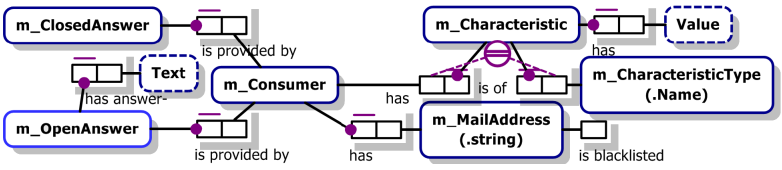

Fig. 5. Part of the optimized ORM conceptual model created as Artifact A1, the 'm_' prefix indicates that these entities refer to data to be migrated

This conceptual model provided abstraction from the implementation choices of the KBM schema and gives the correct semantics of the data. It allowed us to greatly simplify the information consumers' characteristics.

Re-engineering Step 2. During the database analyses we found out that many relations and constraints were implicit in the source schema. To solve this problem, the procedure was done iteratively and together with modeling the source schema in ORM. By closely examining the source schema and the source data, we were able to model all missing and implicit constraints in the source schema in an ORM model.

This ORM model now contains only the entities that must be migrated from the source and all constraints they are subject to. We name this second ORM model artifact A2. To illustrate this result, a part of artifact A2 is shown in Figure 7. This model shows the actual relations between e.g. 'kbm_ProvidedAnswer' and 'kbm_Consumer' that were implicit in the source schema. All constraints that were added are indicated with a model note.

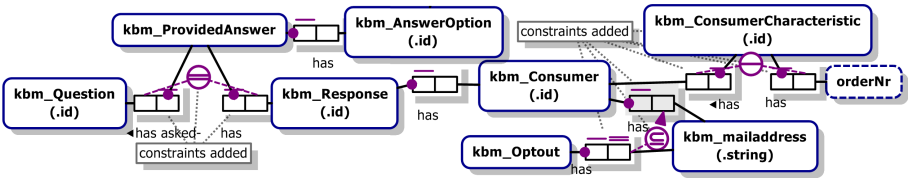

Fig. 6. Part of the ORM Reengineered source model created as artifact 2, the 'kbm_' prefix indicates that these entities refer to the source database

Re-Engineering Step 3. Creating the SQL statements for the first mapping proved to be a complex task. As shown in the previous examples (Figures 3-6), single entities in the target schema contain attributes from multiple tables in the source schema. We used OLE to describe the required mapping between the optimized conceptual model's schema and the unoptimized source schema. OLE is closer to natural language then SQL. Although validating the correctness of the OLE mapping statements is still not a trivial procedure, it proved to support the creation of, and communication about the mapping statements better then writing these directly in SQL statements. Hence it was more easy to validate the OLE with the source system expert.

We used the relational mapping schema resulting from applying the RMap procedure to the artifacts A1 and A2 to construct the OLE statements. Artifact A1 provided the definition of the view that OLE will create. Artifact A2 provides the input from which the data will be extracted.

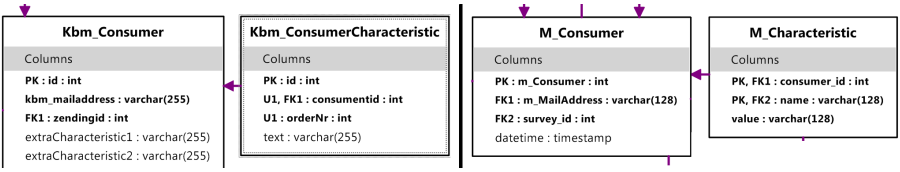

Fig. 7. Part of the ER schema produced from Artifact A2 (left) and Artifact A1 (right)

For each m_Characteristic and Value in m_Characteristic has Value,
(the Value of m_Characteristic is the text of kbm_ConsumerCharacteristic where there is some kbm_ConsumerCharacteristic and consumer_id = consumerid) or
(the Value of m_Characteristic is the extraCharacteristic1 of kbm_Consumer where there is some kbm_Consumer and consumer_id = id) or …
(the same pattern holds for extraCharacteristic 2, 3 and 4).

OLE was translated manually into SQL code that is shown below:

```
CREATE VIEW m_Characteristic AS
SELECT kbm_CCh.consumerid AS consumer_id, kbm_CCh.text AS value, [..] FROM
kbm_ConsumerCharacteristic kbm_CCh, [..]
UNION SELECT kbm_C.id AS consumer_id, kbm_C.extraCharacteristic1 AS value FROM
kbm_Consumer kbm_C, [..].
```

Re-Engineering Step 4 and 5. Step 4 was similar to the re-engineering step 2. The target database also contained implicit constraints. The LS schema did not contain any foreign key constraints. In a test we confirmed that this a potential source of error. Step 5 was similar to step 3, but we found that the target schema contained less quirks than the source schema. The second mapping was still time-consuming, but less complex than the step 3 mapping process.

At the time of writing we are in the stage of the physical data migration. The mapping from the unoptimized source schema to the optimized schema has been implemented as a view. This implementation means that all updates, inserts and deletes on the source database are propagated to the target database server. The next step is to implement the mapping from the optimized schema to the target database schema and to propagate the transactions.

5 Conclusion

This paper described a legacy database migration method that employs ORM's conceptual approach. The ORM approach turned out to be a good implementation-independent way of expressing both domains in the same terms and to learn about the semantics of the model. And it allowed us to split up the mapping process into smaller steps. Despite the small size of our case, we directly experienced the need for a conceptual modeling approach, in order to abstract from the technical details of the source and target system. Future research will have to show that our method will work in different, more complex situations.

In this case, we experienced technical complexities such as a lack of primary keys, implicit constraints stored in the application, duplicate values in the source database, complex relations, conceptual objects represented in different numbers of tables in source and target and different implementation choices. We believe that whichever method may be chosen, these problems will always have to be addressed.

One could argue whether one mapping directly from source to target, instead of two would have been easier to maintain. Even so, the exercise of constructing the two slightly easier mappings may facilitate the construction of an eventual single mapping from source to target.

Problems of scalability are not addressed in this method. We realize that for databases with thousands of tables this method of re-engineering may introduce huge amounts of work and will be practically infeasible.

Future research will entail further validation of the applied method on migration projects that differ in any of the above aspects. Furthermore, creating the view statements that provide the mapping still involves some manual labor. This mapping process certainly will benefit from the further advancement of a natural language-like conceptual query language such as FORML [10], OLE and CQL [11].

References

1. Bakema, G., Zwart, J., van der Lek, H.: Fully Communication Oriented Information Modelling. Ten Hagen Stam (2000)
2. Balsters, H.: ORM Logic-based English (OLE) and the ORM ReDesigner tool: Fact-based Re-engineering and Migration of Relational Databases, Technical Report, Faculty of Economics and Business (May 2012)
3. Curland, M., Halpin, T.: Model Driven Development with NORMA. In: Proc. 40th Int. Conf. on System Sciences (HICSS-40). IEEE Computer Society (January 2007)
4. Curland, M., Halpin, T.: The norma tool for orm 2. In: Pernici, B. (ed.) Advanced Information Systems Engineering. LNCS, vol. 6051. Springer, Heidelberg (2010)
5. Drumm, C., Schmitt, M., Do, H.H., Rahm, E.: Quickmig: automatic schema matching for data migration projects (2007)
6. FBM working group: Fact-based modeling exchange schema. Version 20111021c (2011), `http://www.factbasedmodeling.org/`
7. Halpin, T.: ORM 2. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 676–687. Springer, Heidelberg (2005)
8. Halpin, T.: ORM/NIAM Object-Role Modeling. In: Bernus, P., Mertins, K., Schmidt, G. (eds.) Handbook on Information Systems Architectures, 2nd edn., pp. 81–103. Springer, Heidelberg (2006)
9. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
10. Halpin, T., Wijbenga, J.P.: FORML 2. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 247–260. Springer, Heidelberg (2010)
11. Heath, C.: The constellation query language. In: OTM 2009: ORM Workshop, OTM 2009, pp. 1–10 (2009)
12. Henrard, J., Roland, D., Cleve, A., Hainaut, J.-L.: An Industrial Experience Report on Legacy Data-Intensive System Migration. In: IEEE International Conference on Software Maintenance, pp. 473–476 (2007)
13. Lin, C.Y.: Migrating to relational systems: Problems, methods, and strategies. Contemporary Management Research 4(4), 369–380 (2008)
14. Sluis, T.C.: The ORM Infusion Migration Method, Master's Thesis, University of Groningen (2011)
15. Wu, L., Sahraoui, H., Valtchev, P.: Coping with legacy system migration complexity. In: 10th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2005, pp. 600–609 (2005)