# Data Migration Project Management and Standard Software – Experiences in Avaloq Implementation Projects

Klaus Haller

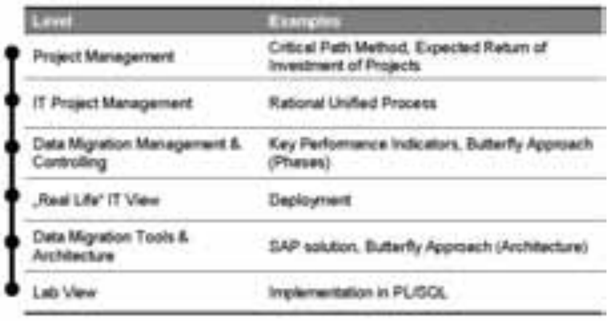
COMIT AG
Pflanzschulstr. 7
CH-8004 Zürich
klaus.haller@comit.ch

**Abstract:** When a bank implements a new core banking platform, it must simultaneously migrate its data from the old platform into the new one. We identified three main dimensions for data migration. They make up the corners in our conceptual data migration project management framework: the *data migration triangle*. The first dimension demands scripts as deliverables, which actually transform and migrate the data. The second, and too often neglected, dimension is the delivery of data. It requires that the scripts run together reliably and on time in a complex environment with many dependencies. Dimension three addresses quality assurance. We describe methods for checking whether all necessary data is migrated and whether the data is migrated correctly. In contrast to previous work, we focus on technical issues from a project manager's perspective. We explain *what* data migration project managers have to manage and *why*.

## 1 Motivation

For some years, a wave has flooded the banking IT landscape in Switzerland. Many banks have replaced their old banking applications with new standard software. They modernized their application landscape and reduced the total number of applications and interfaces. Thereby, the banks cut costs and eased out-sourcing. Avaloq [Ava] and Finnova [Fin] are the two dominant players on the Swiss market [Gab07, Min07]. But implementing one of the two is only the first challenge; the second is the data migration[1]. During this transition, the banks' CIOs and customers share the same fears: The new standard software platform replaced the old platform; the old one is shut down. But later, customers notice that the migrated data is incomplete or corrupt. For example, customers' balances might be incorrect or account statements are mailed to customers who have asked the bank not to mail their statements.

[1] To prevent confusion: *Data migration* means migrating only data out of one schema to a new schema which can be structured completely differently. This is the case when a new application (e.g. Avaloq or Finnova) replaces an old one. Aim of *database migration* project is to change the database version or vendor, e.g. from Oracle 10g to Oracle 11g. Schema, triggers, date etc. should remain unchanged, if it is possible, because applications using the database should remain untouched.



| Level | Examples |
|---|---|
| Project Management | Critical Path Method, Expected Return of Investment of Projects |
| IT Project Management | Rational Unified Process |
| Data Migration Management & Controlling | Key Performance Indicators, Butterfly Approach (Phases) |
| „Real Life" IT View | Deployment |
| Data Migration Tools & Architecture | SAP solution, Butterfly Approach (Architecture) |
| Lab View | Implementation in PL/SQL |

Figure 1: Data Migration Methodology Levels

The key for a successful data migration project is a superior team and a convincing methodology. The methodology in implementation projects is not monolithic but rather like a puzzle. Pieces of the puzzle come from the bank, the implementation partner, the project manager, and some can be chosen by the migration team. This paper offers a new piece of the puzzle: a cohesive lense for data migration project managers to look on their data migration projects, considering not only project organization and specification and implementation, but also the complexity of "real life" IT environments.

Figure 1 provides an overview about the different levels for methodologies relevant for data migration projects. The top level, the project management level, addresses IT and non-IT projects. A good example is the usage of the critical path method or tools like Microsoft Project in companies as well as the return of investment the projects are expected to generate. The second level from the top concentrates on IT specific project management issues like the software development methods (e. g. RUP [Kru04] or Extreme Programming [BA04]). One level down, we reach the management and controlling level for data migration projects. Quite some work has been published in this area. Morris [Mor06] provides an excellent introduction into data migration project organization and project management (though he also addresses underlying technological questions). Shorter articles like [BM04, Hud98] address some of the basic problems and pit falls you might encounter in your first data migration project. The butterfly approach [WLB97] contributes to this level with a high level phase model. Bisbal et al. [BLW97] provide a general discussion of migrating legacy information systems, i. e. the authors discuss more than the data migration aspect. Focusing more on data migration for standard software, [Hal08] provides a more in-depth discussion of the different phases. Also, this paper belongs to the "Data Migration Management & Controlling" level. Based on our experience in different Avaloq implementation projects, our paper describes the tasks of the lower levels with the aim of making their progress measurable.

We name the next level down the "real life" IT view. It includes everything else besides specification and implementation of the migration. Typical topics include integration problems between the data migration sources and all information systems as well as strategies for deploying data or the standard software on servers.

Data migration projects have specific tasks for which they use specific tools (e.g. ETL tools). Furthermore, they rely on a specific architecture. Important previous works contributing to this level include a description of SAP's data migration tools [WG04], a specific language [CG04], a data migration tool [DMS], or the architecture of the butterfly approach [WLB97]. [Hal08] provides a detailed architecture which also comes with suggestions for organizing the scripts. Finally, there is the "lab view" on data migration. The level focuses completely on the narrow aspect of specifying the migration and implementing the specification in PL/SQL scripts[2].

We structure our discussion of the different dimensions of data migration projects by focusing on a management view. Section 2 presents a big picture of the system architecture with specific focus on components and dependencies. Section 3 gives a quick overview on standard software implementation project organization. The data migration triangle, the core illustration and explanation concept, is the subject of Section 4. The triangle provides a high level overview of the three core data migration project dimensions. We elaborate the three dimensions in more detail in the following three sections: Section 5 concentrates on the transformation from the old to the new database schema (level "Lab View"). Section 6 discusses delivery issues, mainly integration and dealing with large amounts of data ("Real Life" IT view). Quality assurance is the topic of Section 7. It belongs conceptually to the data migration and controlling level, but an implementation would belong to the "lab view." It provides a risk model for the technical implementation and how the risks can be tracked. We conclude our paper with a short summary (Section 8).

## 2 System Architecture: Components and Dependencies

Standard software has three core components: a data model including domain value tables, operational data, and workflows (Figure). The *data model* defines the object types managed by the system. In contrast to software development projects, standard software typically restricts changes of the data model. Avaloq, for example, provides the object type "business partner." It must be used for customers, counter parties, etc. Avaloq does not allow adding a new object type "students." However, the data model gives the freedom to introduce a new attribute for "business partners" with values "student," "retired," etc. to model this semantic.

"Student" and "retired" are typical examples for domain value tables. They define a set of values an attribute can take. Another example for domain value tables are bank specific names for account types like "Super Savings Account" or "Senior Residents Checking Account".

Data models offer a data- and IT-centric view of the standard software. *Workflows* implementing business processes represent the business perspective. They specify how users, customers, back office employees, and support staff use the software. Depending

[2] Throughout the paper, we use the term script. Certainly, it is possible to use ETL-tools instead of writing scripts.

on the philosophy of the standard software, workflows can be defined freely, partially, or are completely unchangeable.

Rarely, banks buy standard software for completely new purposes. In most cases, they have been successfully operating for many years. So the bank has already valuable and important data. The old *data* must be migrated into the new system. The data will be complemented by new operational data as soon as the new software is operational.

There are dependencies between the migration of old data, workflows, and the data model. The data model is the foundation for the migration of old data and for the storage and processing of new data (Figure 2 ❶). The data model also influences the workflows (Figure 2 ❷). Workflows manipulate data and depend on data. If the data model does not allow storing a picture of the customer's passport, the execution of a workflow cannot demand such a picture to be stored. Workflows can also put restrictions on data entered by users (Figure 2 ❸) e.g. by only accepting customers whose nationality is known. Depending on the standard software migration philosophy, old data is written directly into database tables or it is migrated using the same workflows as for manual data entry. In the latter case, there is a dependency between workflows and data migration (Figure 2 ❹). Figure 2contains two more components. First, there is an underlying basic infrastructure with hardware, network, databases, and the actual standard software kernel. They are easily overlooked, but our experience shows their major negative impact on the overall project performance if not monitored and managed properly. Furthermore, there is a deployment infrastructure for storing, loading, and copying system snapshots, e.g. for distributing snapshots to different servers. A system snapshot consists of the data model, data, and workflows. Also the standard software kernel or even the complete database might be included. The snapshots are used for rerunning a migration from a certain point (e.g. with migrated customers to test the script for the customers' account script over and over again) or for analyzing problems on different servers. It has not been our experience alone, but others have also observed that there is a huge impact on the development cycle whether it takes ten minutes or ten hours to reproduce a certain situation.

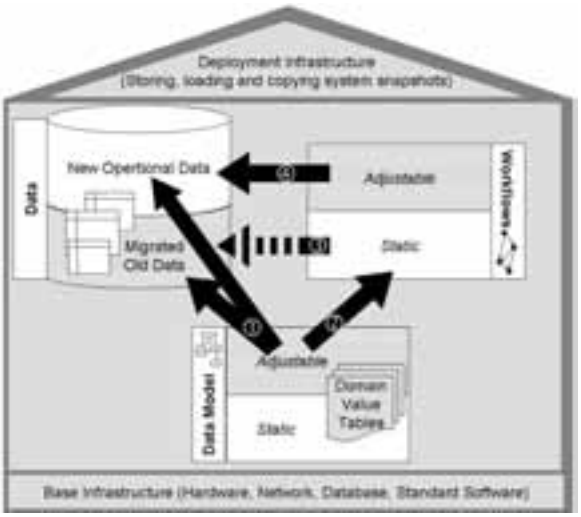

Deployment Infrastructure
(Storing, loading and copying system snapshots)
Data
New Opertional Data
Migrated Old Data
(3)
(4)
Adjustable
Static
Workflows
(1)
(2)
Data Model
Adjustable
Static
Domain Value Tables
Base Infrastructure (Hardware, Network, Database, Standard Software)

Figure 2: System Architecture with Components and Dependencies

## 3 Project Organization

We present a simplified version of the project organization approach of COMIT's Lean Stream Implementation Method [LS07]. We distinguish four teams: a migration team, a customization team, an infrastructure team, and a support team (Figure 3). The *migration team* analyzes the data in the old system, specifies and implements the data mapping for transforming the old data to fulfil the demands of the new system, e.g. implemented in PL/SQL-scripts. Secondly, the migration team ensures the delivery. Delivery means orchestrating script execution and manual data entries. It is a challenging task because even small banks require hundreds of scripts for the migration. The migration team uses specialized migration tools which they manage themselves. Our experience shows that the highly specialized, in-depth knowledge of the migration team is urgently needed, but only until the going live stage of the project. Consequently, our project organization guarantees that the migration team can move to the next project to start over again directly after going live.

The *customization team* is responsible for the workflow implementation and adoption and the domain value tables. Team members usually have a sound knowledge of the business processes and of the standard software in their particular area. The project organization reflects this with a topic-oriented organization like accounting, CRM, etc. in the workflow team. Object modelling, highly important for software development, is a difficult topic in standard software implementation projects. The standard software comes with a predefined object model usually only requiring and allowing minor modifications. However, each data model change might require the migration team to reimplement already developed transformations. Thus, the migration team is highly interested in a stable and consistent object model. Consequently, it gets the responsibility for the object model, but it must collect continuously the requirements from workflow team members.

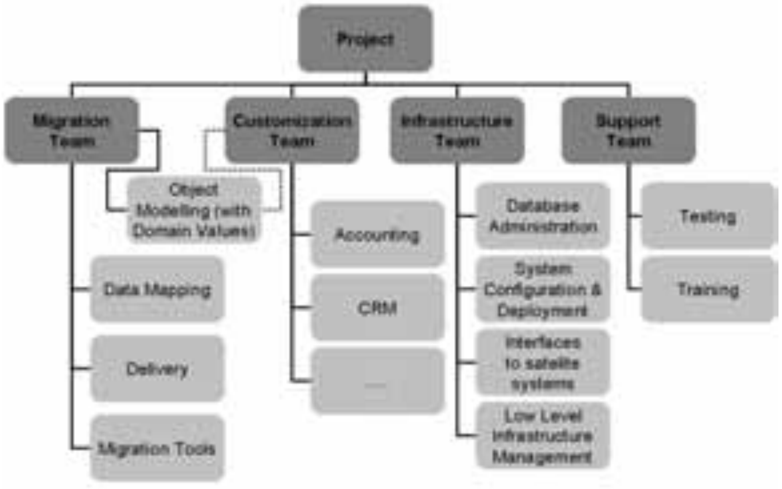


Figure 3: Project Organization

The *infrastructure team* configures and deploys the standard software, implements interfaces with satellite systems like trading, risk, or anti-money-laundering systems, and takes care of basic services such as hardware and network. If tasks like server or database administration are outsourced, the infrastructure team manages the supplier relationships.

The *support team* has two responsibilities. One is at least partially needed after the migration, namely to coordinate the testing with test case definitions, etc. This is also needed for later releases. Secondly, the support team is responsible for the initial user training. Even small banks have hundreds of employees to be trained.

## 4 The Data Migration Triangle

Projects have one thing in common: there are typically more things to be done than staff and time to do them. So it is important to set priorities. Setting priorities requires knowing the tasks. We group all data migration tasks into three dimensions, which are corners of our *migration triangle* (Figure 4). In this chapter, we give a brief overview, whereas we elaborate each task in more details in the following chapters.

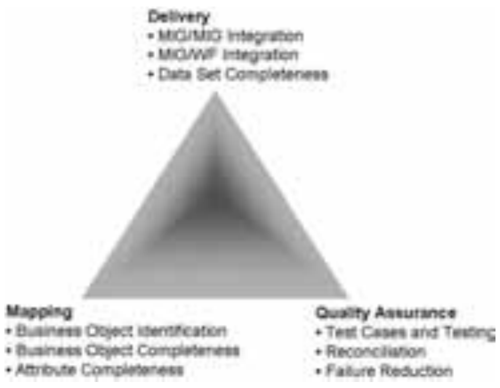


Figure 4: Data Migration Triangle: Dimensions and Tasks

The first corner of the migration triangle represents the dimension called *mapping*. Mapping means defining and implementing the transformation[3] of business objects. A business object is a meaningful object from a user's perspective such as an account, a customer, or a customer's address. The transformation ensures that the data fits into the target platform database schema. The dimension *mapping* has three tasks: the identification of the business objects to be migrated (task *business object identification*), the implementation of transformations for all business objects (task *business object complete-*

[3] We use the term *transformation* for ease of speech in this paper in an inclusive way, meaning that it also comprises the extraction from the old and loading into the new system.

*ness*), and migrating all details of the different business objects (task *attribute mapping completeness*).

The second corner of the data migration triangle, *delivery*, includes correctly executing the different transformations and coordinating them with manual data entries (if not all of the data is migrated automatically). The delivery dimension consists of three tasks: migration-migration-integration (MIG/MIG-integration), migration-customization-integration (MIG/CUS-integration), and data set completeness. The task *MIG/MIG-integration* ensures that the different mapping implementations of the migration team run together. The task *MIG/CUS-integration* takes care that the workflows, domain value tables, and the data migration fit together. The task *data set completeness* tests carefully whether the migration runs in the estimated time for large data sets.

*Quality assurance* is the third dimension. Test cases must be defined and tested (task *test cases/testing*) for ensuring a semantically correct migration result. The task *reconciliation* guarantees that the migrated data is not only correct, but that also all needed data is migrated. If testing or reconciliation detects failures, they have to be solved earlier or later (task *failure reduction*).

The data migration triangle as a conceptual framework enables project managers to express and illustrate priorities in data migration projects and how priorities change during the project's life-cycle. We illustrate this for the three typical situations we identified in our projects: the initial phase, the development phase, and the pre-going live phase. (Figure 5)

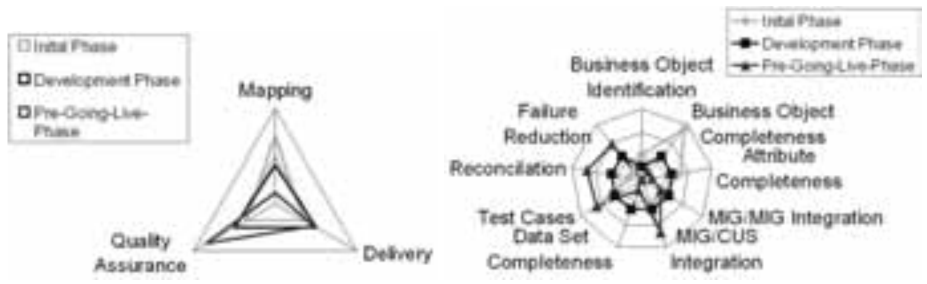


Figure 5: Data Migration Triangle and Extended Data Migration Radar View

Because there are dependencies between business objects, the migration team migrates a basic version of as many business objects as possible in the *initial phase*. Avaloq, for example, allows migrating a bank account only if the customer has been migrated before. Migrating an account balance is only possible if the account exists. Generally speaking, business objects form chains or, more accurately directed, acyclic dependency graphs. The "early" business objects must be migrated early in the project, even if they are not completely correct, such that work on the post-ordered business objects can start.

During the *development phase* all business object types should be looked at in depth, i.e. all attributes have to be considered, and the quality of the migration should be more or less good enough for going live.

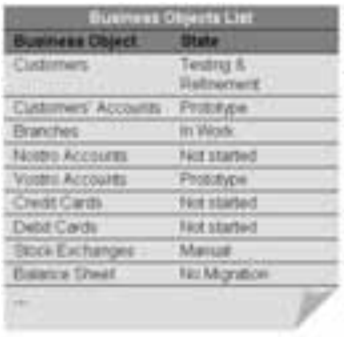
A few weeks before going live, the *pre-going live phase* starts. The migration team must work on ensuring that (nearly) 100% of the objects are migrated automatically. It is especially important that the migration team monitors the data model, all domain value tables, and every workflow change of the customization team to prevent the going live migration from failing due to incompatibilities.

The priorities for the different phases are reflected in the data migration triangle and an extended radar view in Figure 5:

- To reach the goal of migrating as many business objects as possible, priority is given to business object completeness and MIG/MIG integration during the initial phase. Furthermore, test cases should be defined, though they cannot be tested extensively.
- During the development phase, the different subtasks are equally important, if no specific problems arise. The business objects' identification should be nearly completed and require only very limited affords.
- During the pre-going live phase, emphasis is given to a last round of failure reduction, so that not too many manual migration or corrections are required. Furthermore, the integration of the customization and the migration team is important. Systems like Avaloq allow changes of the data model, domain data tables, and migration relevant workflows till the last moment. So the migration team has to run frequent tests of their migration routines to prevent last time changes from causing major problems when going live under time pressure. Certainly, additional test cases are important and must be considered, including retesting previously failed ones.

## 5 The Mapping Task

The most obvious results of the migration team's work are executable scripts for extracting data from the old system, transforming it, and loading it into the new standard software. In a first step, the migration team identifies what shall be migrated (task *business object identification*). The team compiles a list with all business objects to be migrated like the first column of the table in Figure 6. Compiling this list is far from trivial. Theoretically, the migration team can start with an empty list and collect the business objects by interviewing the workflow team, examining the GUI and the current output, etc. However, there is a high risk of forgetting important business objects. We prefer to start with a copy of a previous project and to adopt the copy to the specific needs of the actual bank. The software vendor or experienced consulting companies should provide such a list for their customers.



| Business Objects List | |
|---|---|
| Business Object | State |
| Customers | Testing & Refinement |
| Customers' Accounts | Prototype |
| Branches | In Work |
| Nostro Accounts | Not started |
| Vostro Accounts | Prototype |
| Credit Cards | Not started |
| Debit Cards | Not started |
| Stock Exchanges | Manual |
| Balance Sheet | No Migration |
| ... | |

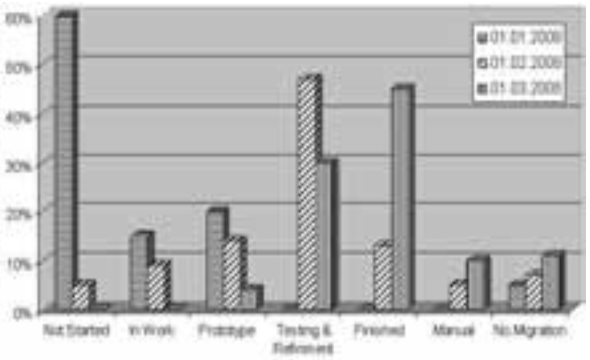


Figure 6: Business Object List (left) an 1

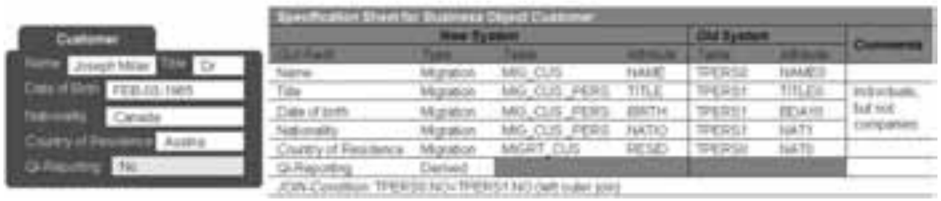
When the migration team has identified at least the most important business objects, the specification of the transformation starts.[4] Our experience shows that a simple mapping sheet is sufficient (Figure 7). Starting points are the different windows of a GUI prompting the different data items of a business object, which we assume to be the most important information to be migrated. The migration team fills out a separate sheet (e.g. in Excel). Each attribute in a window is a row on the sheet. A row stores the information of the attribute's name in the window and the table and attribute name in the standard software. The table might be a table of a migration API or directly the internal table. Furthermore, the table and attribute name of the old system is needed. The migration type states whether the attribute is migrated normally ("migrate") or whether the information is derived automatically by the standard software ("derived" like the QI information which depends on the nationality etc.). It is also possible that no migration is needed ("no migration") or that due to the small amount of data, the data is manually entered ("manual"). If data comes from more than one table of the old system, the join condition between the different tables should be added. After the transformation is specified, it is implemented using SQL, PL/SQL, or an ETL-tool. When a migration run is finished, tests are performed and reconciliation tasks can start.

From a project manager's perspective, it is important to measure the progress of all three tasks. *Business object identification* progress corresponds to the stability of the list of the business objects (Figure 7, left). The key performance indicator is the number of newly added business objects, e.g. per week or month. The lesser new business objects are added, the more stable the list is, and the more likely it is already complete.

[4] To our best knowledge, no well-established graphical specification techniques for data transformation exist (like we know e.g. UML or ER diagrams for data models), though certainly for the technical implementation much research in the context of schema mapping languages was done or is provided by existing ETL tools like [Pow].

*Business object completeness* reflects how many of the identified business objects are already migrated. Our experience shows that instead of a simple “done/to be done” model, a more complex model is appropriate: “not started” (nobody has done anything), “in work” (work has started but no result yet), “prototype” (first version successfully migrated), “testing and refinement” (it works in principle but not every attribute has the right value), “finished” (accepted by the customer). The progress is measured per business object. The overall situation is summed up in a statistic (Figure 7, right) for reporting purposes.

*Attribute mapping completeness* addresses whether all details of a business’ objects are considered. For example, a customer can be modeled in the beginning only by her name.


Customer

|  |  |  |  |
|---|---|---|---|
| Name | Joseph Miller | Title | Dr |
| Date of Birth | FEB-03-1965 |  |  |
| Nationality | Canada |  |  |
| Country of Residence | Austria |  |  |
| QI-Reporting | No |  |  |

Specification Sheet for Business Object Customer

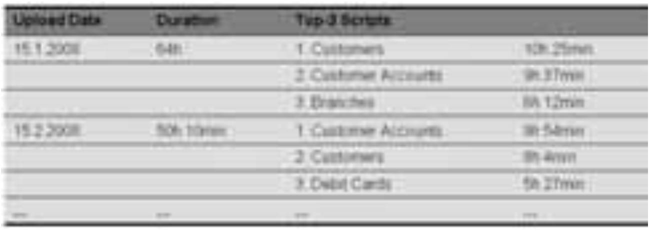
| New System |  |  |  | Old System |  | Comments |
|---|---|---|---|---|---|---|
| GUI Field | Type | Table | Attribute | Table | Attribute |  |
| Name | Migration | MIG_CUS | NAME | TPERS2 | NAME3 |  |
| Title | Migration | MIG_CUS_PERS | TITLE | TPERS1 | TITLE8 | individuals, but not companies |
| Date of birth | Migration | MIG_CUS_PERS | BIRTH | TPERS1 | BDAH8 |  |
| Nationality | Migration | MIG_CUS_PERS | NATIO | TPERS1 | NATY |  |
| Country of Residence | Migration | MIGRT_CUS | RESID | TPERSN | NATB |  |
| QI-Reporting | Derived |  |  |  |  |  |
| JOIN-Condition: TPERS2.NO=TPERS1.NO (left outer join) |  |  |  |  |  |  |

Figure 7: GUI View (left) and Specification Sheet (right)

So the business object exists which is helpful for business objects depending on the existents of addresses. However, all attributes of the addresses must be migrated in the end. Measuring the attribute mapping completeness is not really easy (and counting attributes in the old and new system usually only adds bureaucratic overhead). However, the already discussed statistics for business object completeness also contain this information (Figure 7, right).

## 6 Migration Delivery

When the migration team members have specified and implemented scripts each for themselves, the next two challenges are first, keeping the scripts compatible with the rest of the environment and the other migration scripts, and second, ensuring they run in a sensible time frame. We start our discussion with the latter challenge, the task of *data set completeness*. The underlying problem consists of two contradicting demands. The migration team needs short development cycles. If migrating the complete data of a bank needs several days, the migrated data is restricted for testing purposes to a few representative branches to reduce the migration time and thereby the development cycles. Secondly and contrary, the final migration duration often has an upper limit, because e.g. a bank cannot stop its operation for a week. So the migration team must identify and optimize long running scripts by tests migrations with the complete data set. Especially problematic are transformations with non-linear complexity. If migrating 1,000 customers needs one hour, migrating 10,000 customers might need 10 hours for a complexity of O(n), but 100 hours for a complexity of O($n^2$). Also memory, network, or disk problems might suddenly occur. Thus, it is mandatory to run regular tests. Valuable indicators for the progress of this task are (i) frequency of uploads with the whole data set, (ii) the needed time for completion for migrating all date with all scripts, and (iii) the top-n scripts regarding execution time (Figure 8).



| Upload Date | Duration | Top-3 Scripts | |
|---|---|---|---|
| 15.1.2008 | 64h | 1. Customers | 10h 25min |
| | | 2. Customer Accounts | 9h 37min |
| | | 3. Branches | 8h 12min |
| 15.2.2008 | 50h 10min | 1. Customer Accounts | 9h 54min |
| | | 2. Customers | 8h 4min |
| | | 3. Debit Cards | 5h 27min |
| ... | ... | ... | ... |

Figure 8: Indicators for Task Data Set C 1

The second aspect, *integration* respectively integration tests, can be further structured with respect to the involved teams. There is a customization/customization (CUS/CUS)-integration for testing, e.g. whether the workflow for cashiers for a high withdrawal in a foreign currency works properly with the involved forex dealer's workflow. Frequently, the customization team builds a customization build. It is a consistent set of workflows and domain value tables of all customization team members. The customization build is then rolled out to all customization development servers and the test and release management cycle starts over again (Figure 9).

Of particular interest for us are integration issues involving the migration team. Similar to the customization release management cycle, the migration team performs integration tests by executing the scripts and thereby making a migration build (migration/migration-integration or, short, *MIG/MIG integration*). So they continuously check, for example, whether their scripts fit together with respect to primary/foreign key relationships. The migration release is deployed on the migration development servers. Usually, some migration development servers get an intermediate state after some, but not all scripts have been executed. Developers working e.g. on the migration of customer accounts want a server state where the customers are already migrated, but the customer accounts scripts have not been executed. So they can test improved versions of their scripts.

MIG/MIG-integration and CUS/CUS-integration are two separate release management cycles. Additionally, there are interdependencies between the two cycles ("connected cycles release management", Figure 9), which we term migration/customization-integration or *MIG/CUS-integration*. The migration team must regularly test whether its scripts are still working properly with the newest customization build. For example, if the customization team decides not to offer savings books any more, the migration team cannot migrate saving books unchanged but must transform savings "books" into savings "accounts." Workflows which the customization team adopts are another source of problems. For example, they may add a requirement that the passport-ID is stored for each new customer. If the migration team uses the same workflow for migrating existing customers, they cannot migrate customers for which they do have passport-IDs in the old system. But also the migration can cause problems for the customization team. If the migration team decides unilaterally to not migrate historic interest information, it is impossible for the customization team to generate certain tax reports.

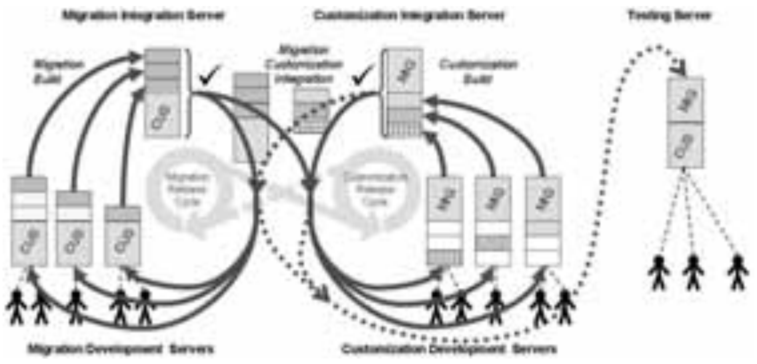

Migration Integration Server
Customization Integration Server
Testing Server
Migration Build
Migration Customization Integration
Customization Build
Migration Development Servers
Customization Development Servers

Figure 9: Connected Cycles Release Management

The dependencies require that the migration team knows the progress of the customization team and vice versa. The *connected cycles release management model* ensures exactly this. The customization team gets a "frozen" migration build from the migration team and the migration team gets a "frozen" customization build. In detail this means that the migration team works on a server with the standard software and a certain customization build (workflows and domain value tables). From time to time, the migration team runs integration tests by migrating the whole data or a subset. In case of a successful migration, the migration server has a new migration build. The migration build serves two purposes: first, it is deployed to the migration development servers as discussed before. Second, the migration test server is deployed to the customization development servers. If the workflows and domain value tables of the migration build are not the newest ones of the customization, value tables and workflows are taken from the last customization build.

Based on the new migration build, the customization team develops and adopts workflows and domain value tables. Then, they run integration tests by installing all workflow and domain value table changes on the customization integration server. This creates a new customization build. This build is deployed not only to the customization development servers. Also the migration team gets a copy as a base for their next migration build, after the already migrated data has been removed. Finally, from time to time, a copy of a migration or customization integration build is deployed to one (or more) test servers, on which testers and end users test the workflows and check the correctness of the migrated data.

## 7 Quality Assurance

*Quality assurance* is the third and last aspect of our data migration triangle. It subsumes three tasks:

1. Migration *test case identification* and their *execution* for semantic verification checks
2. *Reconciliation* for technical migration verification

3. *Failure reduction*, i. e. reengineering scripts which implement incorrect transformations[5]

The tasks indicate differences and common aspects between the classical testing in software development projects and the specific needs of data migration projects. This is what our risk model specific for data migration projects in Figure 10 illustrates. It is based on three assumptions. First, we assume that we do not have to deal with bugs in the old software or the new standard application. Second, we assume that the data can be migrated without prior cleansing.[6] Finally, without loss of generality, the data of the old and the new software store their data in one database. Then, we can group the data migration specific risks as following:

– Semantic failures (Figure 10, no. 1, 2, and 3)

– Syntax errors (Figure 10, no. 4)

– Incompatibility failures (Figure 10, no. 5 and 6)

Semantic failures are conceptual problems during the extraction of data from the old system, during its transformation, or during the process of loading into the new schema. The extract step (1) must identify exactly the needed data of the old platform, which is not always easy. In case a bank migrates its customers to a new system, customers might not be of interest if they died five years ago. But if a customer caused serious problems five years ago, such that the bank decided not to do business with this customer any more, the customer must certainly be migrated and this information preserved. Secondly, the transformation might be wrong (2). Simple failures include mixing up the given name and the surname. A more complex failure would be to put all stocks of customers with more than one custody account into their first one. The third kind of semantic failures appears during the loading (3). This takes place when the migration-API of the new platform refuses certain data or data is written to wrong tables.

[5] From a technical perspective, developing new transformations or reengineering an old one is identical. From a project management point of view it is different. The project manager, as explained in Section 4, might give more emphasis in migrating as many business objects as possible in the initial phase instead of correcting minor transformation failures.

[6] If data cleansing is necessary, there are three options. It can be done in the old system *prior* to the migration. The advantage is that well-trained staff is available and no technical problems due to incorrect data appear during the migration. Data cleansing *during* the migration implies that the transformation solves the quality problems. It requires that all data sources needed for the cleansing are available in the system (often, they are scattered around in many Access or Excel files of different persons). However, the greatest risk is that users and managers take the chance that someone is found for the data cleansing, and then they have more and more cleansing and improvement wishes. Then, the migration team does not get along with its original work. Finally, the data cleansing can be done *after* the migration, e.g. if the project is running out of time. It requires the data quality to be in a good enough state such that it is accepted by the new system and that it does not cause serious operational problems.

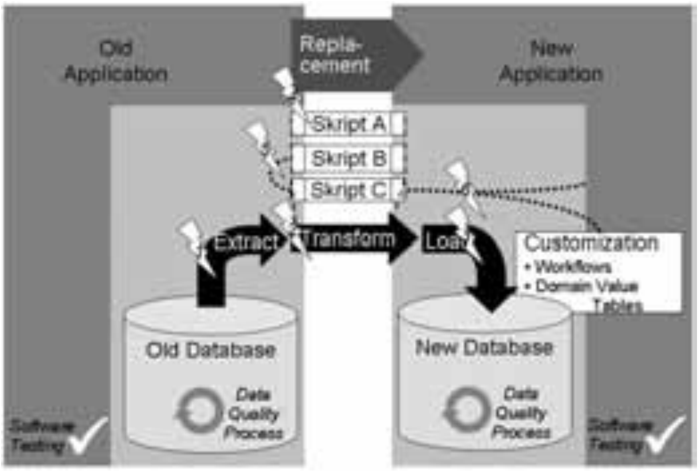


Figure 10: Risk Model for Data Migration

A failure is a *syntax error* (4) if the execution of a script raises an error due to the script not conforming to the language syntax like SQL. Finally, there are *incompatibility* failures between two migration scripts (5) or between a migration script and the standard software and its customization (6). Incompatibility failures reflect problems with MIG/MIG-integration or MIG/CUS-integration. The target platform, respectively the customization, assumes e.g. a key to be unique but the migration delivers non-unique keys. Such failures might raise run-time exceptions or the standard software refuses the data. However, if the standard software does not detect such problems immediately, they might harm stability of the software later.

We require *failure detection methods* for each of the identified risks in Figure together with KPIs as indicators for the change of the risk exposure over time (Table 1). Problems in the *extract step* are best detected using a reconciliation mechanism. *Reconciliation* means checking for each element on the source side whether it is delivered to the target, e.g. whether all customers are migrated. Furthermore, important attributes like nationality or aggregation functions like the sum of the balances of a customer's bank accounts. User tests help in an early phase for detecting if large amounts of the data are missing (e.g. a certain branch or all savings accounts), but are of no use to check whether three out of 100,000 customers got lost. The amount of data items detected as missing (or not existing in the old system) in a certain time frame is a suitable KPI.

*Transformation* failures are detected best by end user tests. A bank counselor could check the portfolio of her most important customers to see whether the data in the new system has been migrated correctly. Data items identified as correct can then be checked automatically in regression tests for later migrations. A reconciliation mechanism is only of limited help, because it does not check details, and it is written by members of the migration team having less experience then users. KPIs are the number of errors during an upload (complete and/or per script) and the quantity of affected data items. Problems during the *load* are also detected by a reconciliation mechanism which might collect the data rejected by the standard software. KPI is the number of unsuccessfully migrated data items. *Syntax errors* should be counted by the person responsible for a test migration. *Incompatibility failures* are more difficult. First, they have to be found if they do not cause a rejection during the load step. So the test migration responsible, or team members of any project team might notice such a failure. He has to report it together with a severity classification.

| *Data Migration Risk* | | *Detection Method* | *KPI* |
|---|---|---|---|
| 1 | *Semantic failures (Extract)* | + Reconciliation<br>o User Tests | Quantity of data set change |
| 2 | *Semantic failures (Transformation)* | + User Tests<br>o Reconciliation | Identified errors and amount of affected data elements |
| 3 | *Semantic failures (Load)* | + Reconciliation | Not loaded items per test migration |
| 4 | *Syntax errors* | + Report of the test migration responsible | Quantity of syntax errors overall/per script |
| 5 | *Incompatibility MIG-MIG* | + Report of the test migration responsible or migration team members | Quantity and classification of severity |
| 6 | *Incompatibility MIG-CUS* | + Report of the test migration responsible, migration team members, or customization team members | Quantity and classification of severity |

Table 1: Failure Detection Methods and KPIs

## 8 Summary

When a bank replaces its core banking application platform with new (standard) software, the data of the old platform must be migrated to the new one. Such data migration projects have three dimensions, which are consolidated in our *data migration triangle*. The data migration triangle expresses and illustrates priorities for the three dimensions and their specific manifestations. In detail, the dimensions are:

– *Mapping* with the task's business object identification, business object completeness and attribute mapping completeness. This dimension addresses that all business object types are migrated with all needed attributes.

– *Delivery* with the tasks MIG/MIG- and MIG/CUS-integration and data set completeness. The dimension represents the fact that the migration scripts not only have to run correctly for themselves, but fit together with each other and the standard software customization. Furthermore, the scripts must be executable within a sensible time frame.

– *Quality assurance* with the task of testing cases, reconciliation and failure reduction. Based on a model for technical migration risks, the dimension cares about tracking the actual risk exposure by looking how it changes over time.

With this focus on explaining the dimensions of data migration projects in companies' complex IT environments, this paper complements perfectly other work on data migration architectures like [WLB97, Hal08] by giving a management's view on the technical aspects of the actual data migration implementation.

## 9 References

[Ava] Avaloq, http:// www.avaloq.com

[BA04] K. Beck, C. Andres: Extreme Programming Explained: Embrace Change, Addison-Wesley, Amsterdam, 2004.

[BLW97] J. Bisbal, D. Lawless, B. Wu, et al.: A Survey of Research into Legacy System Migration, Technical Report TCD-CS-1997-01, Computer Science Department, Trinity College Dublin, 1997.

[BM04] Ch. Burry, D. Mancusi: How to plan for data migration, in: Computerworld, 21.5.2004.

[CG04] P. Carreira, H. Galhardas: Efficient development of data migration transformations, Proceedings of the International Conference on Management of Data (SIGMOD), Paris, France, 2004.

[DMS] Data Management Suite (DMS), www.comit.ch

[Fin] Finnova, http://www.finnova.ch

[Gab07] C. Gabriel: Plattform-Wechsel: Parforce-Übung mit weitreichenden Folgen, in: Schweizer Bank (6/2007), Zürich, 2007.

[Hal08] K. Haller: Datenmigration bei Standardsoftware-Einführungsprojekten, in: Datenbank Spektrum, Nr. 25, dpunkt.verlag, Heidelberg, 2008.

[Hud98] J. R. Hudicka : An Overview of Data Migration Methodology, Select Magazine (April 1998), Independent Oracle Users Group, Chicago, IL, 1998.

[Kru04] Ph. Kruchten: The Rational Unified Process. An Introduction, Addsion-Wesley, Amsterdam, 2004.

[LS07] LeanStream® – COMIT Implementationsmethodik, Version 3.0, Comit AG, Zürich, 2007.

[Min07] M. Minetti: Ein aktuelles Bild der Schweizer Banken IT, www.inside-it.ch, 6.12.2007.

[Mor06] J. Morris: Practical Data Migration, British Computer Society, Swindon, UK, 2006.

[Pow] PowerCenter, http://www.informatica.com/de/products/powercenter/default.htm

[WG04] M. Willinger, Johann Gradl: Data Migration in SAP R/3, Galileo Press, Boston, MA, 2004.

[WLB97] B. Wu, D. Lawless, J. Bisbal et al., "The Butterfly Methodology : A Gateway-free Approach for Migrating Legacy Information Systems", in Proceedings of the 3rd IEEE Conf. on Engineering of Complex Computer Systems (ICECCS'97), Como, Italy, 1997.