

# Testing & Quality Assurance in Data Migration Projects

Florian Matthes, Christopher Schulz  
Software Engineering for Business Information Systems  
Technische Universität München  
Boltzmannstrasse 3, 85748 Munich, Germany  
{matthes,schulzc}@in.tum.de

Klaus Haller  
Swisscom IT Services Finance  
Testing & Quality Assurance  
Pflanzschulstr. 7, CH-8004 Zurich, Switzerland  
klaus.haller@swisscom.com

**Abstract**—New business models, constant technological progress, as well as ever-changing legal regulations require that companies replace their business applications from time to time. As a side effect, this demands for migrating the data from the existing source application to a target application. Since the success of the application replacement as a form of IT maintenance is contingent on the underlying data migration project, it is crucial to accomplish the migration in time and on budget. This however, calls for a stringent data migration process model combined with well-defined quality assurance measures. The paper presents, first, a field-tested process model for data migration projects. Secondly, it points out the typical risks we have frequently observed in the course of this type of project. Thirdly, the paper provides practice-based testing and quality assurance techniques to reduce or even eliminate these data migration risks.

**Keywords**—data migration, process model, risk, testing, quality assurance, risk mitigation

## I. MOTIVATION

Regarding information technology (IT) maintenance, more than ever companies are confronted with the challenge of migrating data from at least one source to one target business application [1]. Thereby, data migration is understood as a tool-supported one-time process which aims at migrating formatted data from a source structure to a target data structure whereas both structures differ on a conceptual and/or technical level [2]. While the first level refers to business object types which are meaningful conceptualizations of objects in the business world (e.g. contract, product, customer account), the second level denotes their technical realization within the databases.

Reasons for initiating a data migration project are manifold:

- Corporate events like mergers and acquisitions (M&A) or carve-outs lead to company-wide data integration and consolidation endeavors [3], [4].
- Implementation of novel business-models and processes brings along new functional and non-functional requirements no longer supported by the existing application. Its replacement induces the migration of the contained data [3], [5].
- Technological progress and upgrades, e.g. migration to more commoditized platforms or off-premise data storages [6]–[8].
- New statutory and regulatory requirements demand for adapted business processes only supported by up-to-

date business applications. Decommissioning the existing application comprises the migration of data, too [9], [10].

Above drivers entail the re-occurring replacement or consolidation of existing business applications as a type of maintenance. As a consequence, data migration projects represent an everlasting although infrequently performed discipline [11], [12]. Given their one-time event nature coupled with a permanent underestimation in size and complexity [13]–[15], existing know-how and experience is sparse and not subject to a diligent documentation process. In this spirit, it is not surprising that the research group Bloor, among others, reports on a “[...]current success rate for the data migration portion of projects (that is, those that were delivered on time and on budget) is just 16%.” [1]. As a main reason, consulting companies like Endava emphasize that “Few companies have the necessary skills to manage, build and implement a successful data migration.” [3].

Modern companies consider their data as a valuable asset. However, any unplanned and crude movement of this asset in the shape of an unprofessional migration project exposes that company to a higher risk [16]. It is therefore vital to follow a stringent and stepwise approach which directly addresses data migration risks. By delineating a practice-based risk model, this paper contributes to the topic of risk mitigation in data migration projects. Furthermore, it describes how these risks can be addressed via a combination of

- 1) a comprehensive data migration process-model elaborated and refined in the course of several industry data migration projects, and
- 2) a set of dedicated testing and risk mitigation techniques coping with the errors from a technical and organizational perspective.

Despite the fact that there are other mature techniques aiming at persisting data (e.g. object oriented, network database systems), relational databases are still widely employed in a business and end-user context [17]. Against this background, the paper focuses on the migration to relational data structures. Moreover, it examines the case the authors of this paper coped with the most, namely where data from one source database has to be migrated to exactly one target with no data.

The remainder of the paper is structured as follows: After having provided a crisp overview on related work in the field in Section II, this paper explains the core elements of any

data migration project (Section III). Based on a practice-proven process model (Section IV) encountered risks are compiled and structured by means of a risk model in Section V. Whereas Section VI explains important technical testing techniques mitigating the priorly identified risks, Section VII elaborates on organizational measures also helping to bring down these risks. Finally, Section VIII concludes with a short summary and outlines the future work.

## II. RELATED WORK

Data migration is a practitioners' topic, recent publications originate mainly from the consulting and tool vendor community. This section sheds light on work published in the area of data migration and distinguishes it from this contribution.

A comprehensive resource to plan, organize, implement, and steer through data migration projects is provided by [18]. From a project manager perspective, the author provides guidance for practitioners confronted with a data migration challenge. Occasionally, project management risks (e.g. too many parties, late sign-off, big bang implementation) are described by Morris who considers them as stumbling blocks better being avoided. Shorter articles (e.g. [6], [16]) target the same audience, discussing very basic problems and pitfalls in the realm of data migration projects. The task of risk reduction is discussed very briefly by this sources emphasizing that in the majority of cases a failed data migration project leads to an unsuccessful target application replacement. Neither a systematic model nor a dedicated technique are presented by the above sources.

When it comes to concrete process models and methods tailored to data migration projects, the butterfly approach [19] has to be given a closer look. Divided into 6 distinct phases, it leverages the concept of temporary data stores in order to allow for a step-wise migration. However, the main emphasis is put on the technical movement of existing and continuously created data. Additional activities like cleansing, in-depth testing, or risk mitigation is only mentioned but not further discussed. Publications like [3], [14] issued by research and consulting companies likewise concentrate on the process of data migration. In elaborating on the different phases a project has to go through, these sources provide a good overview on the necessary activities, key deliverables, as well as roles, responsibilities, and best-practices. Nonetheless, the topic of risk is coped with on a very general level only. Furthermore, suggested testing measures are not explicitly linked with the risks they attempt to mitigate.

Tool descriptions primarily focus on how to leverage proprietary software (often target application specific tools) for data migration projects. For instance, [12] presents a method to migrate data to a SAP ERP application by deliberately avoiding the coding of one-time migration logic. Even if the authors point out some risks (e.g. personnel bottlenecks, project underestimation), they do not suggest any explicit techniques to address them. Further tool-centered literature explains how to successfully migrate to a vendor-specific target application leaving out concrete techniques to address risks.

The process model this type of literature applies is either tailored to the target application (e.g. [20], [21]) or the software tools (e.g. [10]) the commercial vendor offers.

Above discussed literature falls short in explicitly identifying and addressing risks arising in the context of a data migration project. Nonetheless, it lays a solid foundation for the data migration process model and quality assurance measures described later on.

## III. DATA MIGRATION ARCHITECTURE

All data migration projects implement migration programs, scripts, and transformation rules. The programs handle and modify the data, e.g., savings accounts or car orders. They differ from industry sector to industry sector, from company to company, and from project to project. Nevertheless, they rest upon a generic data migration architecture. For this reason, the underlying ideas and principles guiding through the project - and even concrete technologies and frameworks - can be re-used in projects throughout all industry sectors [9].

First, the general structure for migration programs is generic. So are the concepts of an orchestration component and of the (three) data migration databases (Figure 1). These databases are:

- The *source staging database* stores a copy of the source application databases to uncouple both databases. This prevents the data migration project from harming or slowing down the still-in-use source application database.
- The *transformation database* stores intermediate results of the data migration programs. The migration programs join, compare, aggregate, filter and transform large amounts of data. This should never be implemented in Java or similar languages. Databases are optimized for handling large data sets efficiently, e.g., when to execute operations in main memory and when to write data to disk.
- The *target staging database* stores the result of the transformation which is ready for the target application upload. It depends on the target application whether there is an upload API or whether the data must be copied directly into database tables. In the latter case, the upload staging database structure contains tables equivalent to the ones of the target application database. If there is an upload API, the tables in the target staging area contain invocation parameters for invoking the migration API.

Certainly, above databases can be understood as logical instances. Thus, one physical database instance can store all three logical ones using, for example, three different schemas.

The term *data migration programs* refers to the programs moving the data from the source application database to the target application database in transforming the data and its representation as needed. The singular term *data migration program* subsumes three programs needed to migrate one business object type (cf. Section I): the extract & pre-filter program, the transform program, and the upload program. Even if the programs are project-specific they follow a generic structure. This eases setting-up projects and dealing with

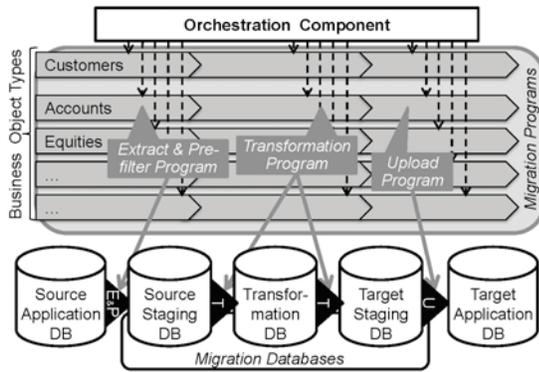


Fig. 1. Data migration architecture

changes in staffing. Two dimensions characterize the programs: the stages and the business object types.

The first stage is pre-filtering and extraction. A *pre-filters & and extract* program copies the relevant data from the source application database. The data remains untouched to the greatest possible extent. Pre-filter means to reduce the data but not to decide upon which data must be copied into the target application database.

For instance, the exclusion of accounts closed decades ago. However, specific internal accounts would not be filtered in the pre-filter phase. This needs an in-depth analysis to decide whether they are needed in the target application. Enriching the data with data not originally stored in the source, but required by the target application [22], [23] can take place now or in subsequent steps, the *transformation program*. The transformation program contains all data mappings between source and target structure. For instance, the account type ID for savings accounts might be 4553 in the source application and 85571 in the target. The transformation program also performs the fine-granular filtering to decide which data the target application needs. The third and final program is the *upload program*. It loads the data into the target application database. If provided, the upload program can use a target application API or write the data directly into the tables of the target database<sup>1</sup>. Executing all data migration programs in the correct order is a challenge. Thus, an *orchestration component* ensures the correct starting order of the programs using a timetable-like mechanism.

#### IV. DATA MIGRATION PROCESS MODEL

The main objective that all migration projects have in common is to permanently move the data from the source business application into the target in implementing a specific process. This paper is based on a process model the authors repeatedly applied in practice. It consists of four main stages which in turn contain fourteen distinct phases. The model, depicted in Figure 2, is described in [2] in greater detail. Its main stages are:

<sup>1</sup> [11] discusses the various options in detail.

- *Initialization*, i.e., setting up the necessary organization and infrastructure
- *Development*, i.e., developing the actual data migration programs
- *Testing*, i.e., validating the correctness, stability, and execution time of both, data and the data migration programs
- *Cut-Over*, i.e., finally switching to the target application by executing the migration programs

Subsequent to the cut-over, the data migration project is finalized and the productive operation phase of the new application starts.

In line with our experience, this section focuses on the actual process of a data migration project including key deliverables. The process forms a solid foundation allowing for risk identification and discussion of appropriate mitigation techniques in subsequent sections. Deliberately, the testing stage and its phases are discussed later in this paper after most relevant risks have been identified and explained. From time to time, pointers to related literature are provided allowing the reader to delve into the topic of data migration.

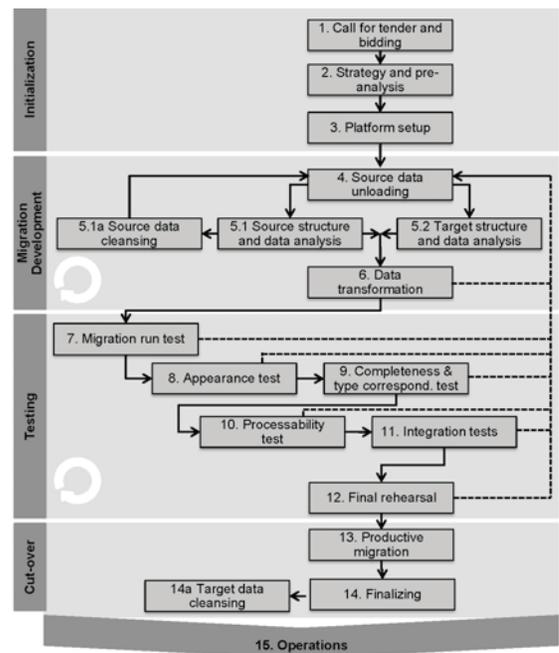


Fig. 2. A process model for data migration

##### A. Initialization

Before the first developer can work on the data migration program, the project organization and the technical infrastructure have to be established. This is achieved in the initialization stage which consists of three distinct phases: call for tender and bidding, strategy and pre-analysis, and platform set-up.

**Call for tender and bidding phase:** Throughout this phase the company determines who will carry out the data migration project. It can be assigned to the internal IT department or a third party company. There are three main reasons for outsourcing this type of project. First, a data migration project

requires additional time and effort which has to be performed on top of the ordinary IT tasks of plan, build, and run [12]. Unfortunately, this type of project is not business-enabling and the necessary time and costs are often underestimated [1], [11]. Secondly, it takes time and effort to build up internal competencies. Thus, keeping an internal team is only reasonable for large companies (e.g., global acting financial institutions, international car manufacturer) or for companies offering this service to customers. Thirdly, data migration projects are too infrequent (approx. every 5-10 years) for building up and foster internal know-how. Since a delayed data migration project delays the overall business application replacement project it is therefore advised to make use of an external migration provider. The key deliverable of this phase is the bidding report. It contains the scope, affected business applications and stakeholders, necessary resources, project plan, as well as anticipated risks with regards to the project.

**Strategy and pre-analysis phase:** Main tasks of this phase consists in the elaboration of the project scope and a migration roadmap. The latter has to cope with fundamental migration strategy decisions such as big bang cut-over versus an incremental migration approach [24]. The roadmap must also point out what to do with historic data residing in the source application. Further possible challenges must be addressed and decided on, e.g. a large data volume, a target database already containing data, organizational challenges such as distributed project teams, or poor data quality. Depending on the time and the budget, a small test data set can be prepared and migrated as a proof-of-concept to convince the customer and to increase project moral(cf. [2]).

**Platform setup phase:** This phase establishes the technical development and execution infrastructure as alluded to in [14], [22], [25]. Core part of a data migration project consists in the development of transformation programs for permanently moving the data from the source application's data structure into the new one. Therefore, databases for intermediate storage must be set up (so called source and target staging area), a repository for the migration program has to be prepared, and specific data migration tools must be installed.

## B. Development

The development stage covers all relevant aspects for implementing the data migration program.

**Data unloading phase:** In this phase, the project team prepares a fresh snapshot of all relevant data by developing and using the pre-filter & extract programs. As a result, the relevant data from the database of the source business application is copied to the source staging area.

**Structure and data analysis for source and target:** It is vital to learn as much as possible about the data and its structure of both-source and target database- [3], [15] before and during the implementation of the transformation programs for the various business object types. For an in-depth analysis it is required that the data structure of source is mapped to their respective staging area and that the data of the source has been unloaded. Obtaining an understanding of all necessary

tables and attributes in addition to a data quality assessment can turn into a tedious and time-intensive task which makes the balance between pro-active analysis work and deferred testing indispensable. Final outcome is the data analysis report, a document which details on the migration-relevant source data with respect to its quality and structure at the attribute level. The deliverable also comprises action items with regards to an optional data cleansing phase including an estimate of the associated costs and necessary resources. Likewise to the source, the target data structure eventually has to be mapped to the respective staging area to allow a thorough examination of the structure [7]. Fortunately, the analysis of source and target can be performed in parallel helping to speed up the overall migration process.

**Source data cleansing phase:** As a matter of fact, data migration projects often ferret out data quality issues given the legacy nature of data involved [14]. The data cleansing phase aims at correcting corrupt and arcane data leading to an improved data quality. In general, cleansing should take place either in source or target database but may be also part of the unloading and transformation activities(cf. [3], [9]). Scrubbing the data in the source database separates this task from the analysis of the target structure. This becomes advantageous especially if development and configuration work on the target application has not been finished yet. In this vein, future errors being caused by unclean data can be circumvented [7] since the data is rectified upfront. Alternatively, the data can be cleansed when being unloaded, transformed, or imported from and to the different databases and staging areas. However, this would mean to burden data migration programs with additional cleansing logic, hence adding complexity to the code routines which are supposed to efficiently shift data from one database to another.

**Data transformation:** As for the unloading and analysis phase, the transformation phase is repeated multiple times accounting for an incremental and iterative approach of migrating data (cf. [3], [14], [24]). For each iteration, the data transformation programs containing the rules and logic to migrate the data from the source to the target data structure is gradually extended and continuously improved. The ultimate goal is the complete and correct migration of all relevant data from source to target staging area. In doing so, the concept of transformation business and technical transformation rules as described in [2] is employed. Unfortunately, the mapping of primary and foreign keys from source to target databases might turn out to be an intricate and complex task (cf. [5]). For different reasons, not all migration-relevant data are covered by transformation rules. If this is the case, manual entering, non-migration, or cleansing might facilitate the treatment of these outliers.

## C. Cut-Over:

The cut-over stage is subdivided into three phases: productive migration, finalizing, and an optional data cleansing taking place in the target application.

**Productive migration:** Before finally executing the data transformation program, the data residing in the source application has to be frozen [7], [18]. A final approval meeting is summoned where all project team members decide in favor of or against the import of the data in the target database. A “go-decision” for the migration leads to the execution of the data migration programs including the loading of the target database with data from the target staging area. The point-of-no-return is reached when the target database is released into production, given that a fall back to the source is almost impossible or at least very cost intensive <sup>2</sup>.

**Finalizing:** The finalizing phase is initiated at the moment the target application goes into production with the migrated data and active end users. Thereby, a clear definition of “done” allows for a smooth transition of responsibility from the migration team to the IT department [3], [14]. Additionally, a well-planned and implemented handing over period ensures adequate performance, data quality, as well as end-user acceptance of the target application. Furthermore, an established monitoring mechanism helps to discover areas of improvements while a short report containing experiences and lessons learned facilitates future data migration projects [8].

**Target data cleansing:** If the data was not cleansed in the source application or in conjunction with the unloading and transformation, a data scrubbing should take place at the time the target application is in production [3]. Main driver for a deferred cleaning process is the overall delivery speed, which drops if the data is cleansed along the migration process. However, procrastinating data cleansing often leads to a lack of confidence and integrity regarding the target application and the data contained therein.

## V. RISK MODEL

Data migration projects are often highly visible within the organization. The authors noticed often a latent fear that the switch to the target application or the data migration might fail. Our data migration risk house (Figure 3) helps to move on from vague fears to concrete risks. It complements generic project management methodologies with data migration specific risks. It has three levels: the business level, the IT management level, and the data migration level. In the following, each level is explained and backed by examples.

The top level contains the business risks, often articulated by the customers<sup>3</sup>. The three most relevant business risks are:

- *Profitability* is about costs and income. This covers, first, the direct costs. The costs of the data migration project itself must stick to the project budget. Second, there can be indirect costs of wrong data transformations. For instance, a car manufacturer might have orders for 500 red and 20’000 black cars. If, after the migration, there

<sup>2</sup>Depending on the specific project needs, it is also possible that the productive migration takes place in more than one step (e.g. master data on the first weekend, bookings on the second). [26] elaborates on these questions in more detail.

<sup>3</sup>Business customers can be CEOs of banks in case of a bank merger. It can also be a production line manager when a factory implements a new order management system.

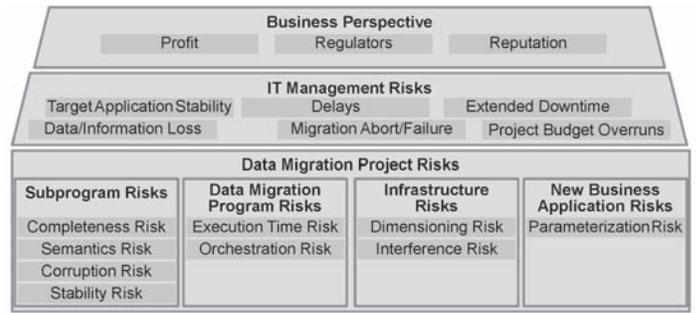


Fig. 3. Data migration risk house

are orders for 20’000 red cars and 500 black cars, there are thousands of cars assembled wrongly. Third, there are also follow-up effects. Data migration projects as part of a system transition project can be the precondition for new offers, lower operations costs, or higher revenues.

- *Reputation:* If a bank ‘disregards’ to migrate 20 saving accounts with CHF 5, fixing this mistake is inexpensive. But if the 20 saving accounts result in negative media coverage, this hampers the reputation of the bank.
- *Regulation:* In certain sectors, e.g., the financial industries, regulators are institutions to be respected if not feared. They must never obtain the impression that the IT system is not managed properly. Otherwise, monetary penalties are not the most severe form of punishment.

The second risk group comprises important **IT management risks**. They have a technical focus and are on a high level.

- *Data or information loss.* The data migration must get all relevant data from the source application into the target application without changing their semantics.
- *Target application stability.* Even if all relevant data got into the target application, there remains the risk that the migrated data harms the target application stability, e.g., if not meeting implicit assumptions about the data structure.
- *Cut-over aborts.* The cut-over does not succeed due to data migration errors. This can be aborts of the data migration programs. Alternatively, the programs run successfully, but the data does not allow to switch to the target application.
- *Extended downtime.* The cut-over needs more time than planned. For instance, the production line or the core-banking system are not available longer than expected. Thus, the bank cannot provide its services and the production line cannot produce cars.
- *Project budget overruns.* The IT department and IT managers have budgets and have to report budget and actual costs. Of course, higher project costs bring them into uncomfortable situations.
- *Delays.* Simple project delays (without any extra costs), can be a risk. Delays can result, for example, in non-availability of key-staff for the cut-over, expiring licenses of the old application, or interface-incompatibilities if tax

authorities or stock exchanges shut down old interfaces

As our experience from project proves, the data migration project manager needs the trust of the business and IT managers. Thus, he must be prepared to answer questions raised by business regarding business and IT management risks. However, these are abstract risks. The data migration project manager cannot address them directly. Instead, he must show how business and IT management risks map to concrete data migration project risks. Then, he can point how he address the data migration project risks he addresses and how they mitigate the higher-level risks.

The **data migration project risks** fall into four categories:

- *Data migration program risks* cover all risks associated with data migration programs. The following list depicts the most important ones:
  - The *completeness risk* covers the risk of losing one or more business objects during the data migration. It subsumes, too, the risk that additional business objects (such as orders for blue cars) appear in the target application that did not exist in the source application.
  - *Semantics risks* can appear if a business object has been migrated (e.g., a car order or a saving keeping account), but the semantics have changed. The order was for a blue car, but now it is a red one. It can also be that the value remains the same but the unit changes. The source application might be working with USD, the target one if EUR. So 500 USD must not become 500 EUR.
  - *Data corruption risks* means that the migrated data does not reflect the data model of the target application. This can happen if there are constraints on application level (e.g., that only bookings on savings account are possible that does not result in a negative account level) which are not enforced by the database. The database does not prevent the loading of 'corrupt' data, but the application might crash later on due to this.
  - The *stability risk* is the risk that coding errors result in a crash or unintended abort of a data migration program.
- The *migration run risks* refer to the execution of the complete set of data migration programs. There are two types:
  - *Execution time risk* reflect the possibility that data migration programs run longer than expected. This can happen for example if all tests were done with a data subset. If 10% of the data needed one hour to be migrated, it does not mean that 100% is finished within ten hours. If sorting or joins are involved, the linearity assumption might not hold any longer. If the database can process 10% of the data in main memory without disk access, there is no guarantee that the database can also handle 100% of the data in main memory.

- *Orchestration risks* address the challenge of coordinating all data migration program and invoking them in the right order. Even migration projects for small- and mid-sized companies may encompass hundreds of data migration programs. Not a single one should be forgotten. Nor can the order be changed easily due to dependencies between business objects. This is the reason why the orchestration task should be automated. Despite automation, the risk remains that not all data migration programs are executed in the correct order.

- The data migration programs and the orchestration component rely on an infrastructure. The *infrastructure risks* refer to risks associated with the technical infrastructure.
  - *Dimensioning risk* reflects unreliabilities under high load. This can be on the network layer, the storage, or even the database layer. To give a concrete example, the undo log could overflow because migration programs have different characteristics than normal applications.
  - *Interference risks* appear if not only the data migration team is active on the source application during the cut-over, but also users, developers, or batch processes. If, for example, one user access the table storing all customers, this can result in a table lock. A table lock during the cut-over would impede the data migration of the customers, and, thus, blows-up easily the complete data migration project.
- *Target application parameterization risks* are risks originating from the target application. If the target application changes, it can become incompatible with the data migration programs. For instance, if the target application parameterization is changed such that it does not allow to add customers without proper identification, this might reflect today's regulation. However, if customers and accounts exist in the source application they must be definitely migrated. The interplay between data migration and target application parameterization may possibly lead to problems.

The remainder of the paper focuses on the risks the data migration projects can influence directly: project risks.

## VI. TESTING-BASED QUALITY ASSURANCE

Quality assurance aims at finding faults in the data, data migration programs, and the underlying infrastructure. Quality assurance has two aspects: the organizational and technical ones. While former correspond to project management issues discussed in following section, latter ones are addressed by testing. This section elaborates, first, testing techniques for detecting errors, second, the risks addressed by each technique, and, third, when how testing is accounted for by the data migration process model.

### A. Testing Techniques in Data Migration Projects

There are two categories of testing techniques (Figure 4): migration run tests and data validation. Latter are further

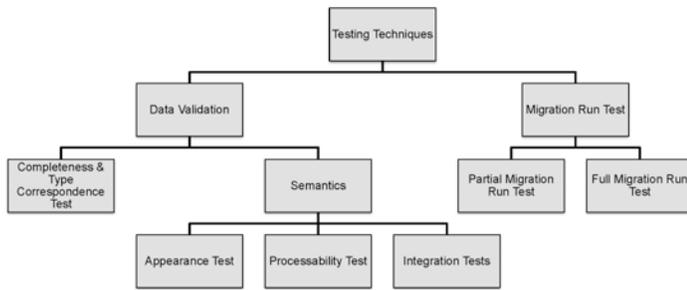


Fig. 4. Overview on testing techniques in data migration projects

grouped into the dimensions semantics and completeness<sup>4</sup>. **Data validation** addresses four aspects: completeness, semantic correctness, consistency on the structure and the data level, and interoperability with other applications. Thus, a sophisticated combination of automated and manual comparisons has to be applied for validating the data and its structure [11]. The main challenge is to compare data of two-source and the target-databases, with respect to predefined comparison criteria [3], [25].

**Completeness & type correspondence tests** attempts to identify business objects that are missing in the target database (e.g., a customer that was not migrated) or new objects (e.g., a customer in the target application who did not exist in the source application). Such tests, typically named reconciliation [11], look at thousands of business objects demanding for automating the comparison of the business objects in the source and the target database. Reconciliation scripts automatically map primary keys (respectively identifiers) of the source and target (reconciliation) to ensure completeness. The reconciliation is the only test covering *all* data. Thus, it is important to validate whether the business object types remain unchanged.

The mapping between source and target IDs is challenging if the IDs or the data structure change during the data migration. Then, the reconciliation must imitate these changes. The imitating code should not be copied from the filter & transformation programs that are tested by the reconciliation. Instead, fresh code must be developed to not repeat past conceptual and implementation errors [9], [11]. Key deliverable of the completeness and type correspondence tests are, first, a list with all business objects missing in the target application and, second, a list with all business objects that are in the target application though they should not be.

**Appearance tests** focus on the appearance of objects on the Graphical User Interface (GUI) level. Testers from the business domain manually compare objects by looking at the GUI data of the source *and* the target application. Depending on the specific business domains and their timely constraints, the data migration team can attempt to automate some comparisons tests to save time. However, there is a risk if automating

<sup>4</sup>The authors rely for the categorization on our work on correctness of information system upgrades [27]. They are directly applicable to the needs of testing data migration programs.

appearance tests especially if the *context* of data changes. An (over)simplified example is comparing the account balances: if the number is the same, but the currency changes, this error is most likely overseen by reconciliation tests [27].

When planning an appearance test, individual business domain experts identify a concrete sample set of business objects for testing. They know best which business objects and data constellations are most crucial [18]. This ensures a high coverage while comparing only a limited number of business objects. If future user conduct the tests, they even get acquainted with the target application they work with in the future. The deliverable of an appearance test is a statement for each business object of the test set whether it was migrated correctly.

**Migration run tests** shall ensure a well-working and smooth orchestration of the data migration programs. They validate the data migration programs in their entirety: pre-filter and extract, filter and transform, and, finally, the load into the target application [3], [25]. The two subclasses of migration run tests, full and partial, focus on different aspects. Full migration run tests run all migration programs with the full data set. This allows to measure the execution time of the overall data migration being identical for the final cut-over (if the infrastructure remains the same). The execution time influences directly the downtime of the application, i.e. the interruption of the business operations. Moreover, a shorter downtime leads to a higher number of migration options when to migrate (e.g. a two hours data migration can be done over night, whereas a 24 hour migration requires longer holiday periods such as Christmas). If necessary, the data migration programs' code is re-visited aiming at improving the migration time.

Partial migration run tests speed-up the round-time of the development cycles during the migration development. Migrating less business objects means delivering results of a trial migration quicker, e.g. within a few hours instead of three days. This eases the development, especially in the beginning. On the negative side, there is a higher risk of inconsistencies in the migrated data. For instance, if only 1% of the customer accounts are migrated, most of the migrated balance sheet data and internal accounts are not consistent with the customer account data. Also, rare data combinations can get lost.

The deliverables of a migration run test are the logs with all error messages, crashes, and—in case of a full migration run test—the needed execution time.

**Processability test** [27] (or mock-load [16]) ensures the coordinated and successful interplay of the target business application and the newly imported data [3]. Whereas the appearance test examines objects at the source and target application GUIs (visual comparison), the processability test processes migrated data. This helps identifying inconsistencies and incompatibilities between the migrated data and the parameterization of the target application. To give an example: the data migration writes the data directly into database tables and the database schema does not enforce that the currency is set. As a consequence, when checking the migrated object

at the GUI, everything looks perfect. But if one processes the data, the application crashes. These errors are detected only when executing concrete test cases. It helps when these test cases are derived from the business processes supported by the application.

Also vital for processability tests is to define sample subset of business objects to ensure a high coverage with a limited amount of test cases [2], [27]. Again, our experience shows that the future users should test for familiarization purpose. In addition to manual testing, application specific batch processes should be prepared and executed. In case of core-banking systems, examples for important batch processes would be host jobs for end-of-day or end-of-year processing and important business events such as opening balance, final account, or insurance rate calculation (cf. [28]).

In most companies, a business process is not supported by one single application but spans over various applications. This demands for end-to-end test or **integration tests**. If one application changes, its interplay with the other applications must be tested. The replacement of a source application with a target application together with a data migration represents such a change. Hence, the proper functioning of the target application with the migrated data in context of its interlinked applications has to be tested. Originally, business objects stored in the source application might have been referring to business objects in other applications. In turn, connected applications were accessing different business objects stored in this source application. The object of integration testing is to check whether the references between applications are still well-working in both directions, e.g., by executing processes spanning over several application. Integration testing is also a techniques addressing the semantics, but now broadens the scope from one application to the overall application landscape.

### B. Mapping Testing Techniques to Risks

The testing techniques aim at mitigating data migration risks grouped in our migration risk house (Figure 3). The first subprogram risk, the *stability risk*, is about potential crashes of a migration program due to bad code including unthought-of and rare data constellations. A partial migration run test covers the non-data constellation risks, whereas all data-specific risks require a full migration run test to ensure that also rare data constellations are covered.

The *corruption risk* reflects discrepancies between the database schema and the data model implicitly assumed by the application code. The suitable testing techniques are appearance tests (validate if the data on the GUI looks non-familiar in the target application; no comparison with the source needed), processability tests (check if the application crashes when performing business processes within the application), and integration tests (look for crashes in the target application or connected system when processing business processes spanning several applications).

The *semantics risk* focuses on whether the data semantics changed during the data migration. The appropriate testing

techniques to mitigate these risk are similar to the ones addressing the corruption risk: appearance testing, processability testing, and integration testing. The difference is that testing for corruption risks checks only for crashes or strange looking data (smoke test-like), whereas mitigating semantics risks demands for comparing GUI data and business process results of the source and the target application.

The *completeness risk* requires solely to run the scripts of the completeness and type correspondence tests. The data migration program risks *execution time risk* and *orchestration risk* and the infrastructure risk *dimensioning risk* can be addressed by performing full migration run tests. When all data is migrated and all migration programs succeed, the execution time can be measured. This also provides an answer to the question whether the hardware and system configurations cope with the full data set. Only when it comes to the orchestration risk, also a partial migration is sufficient *if* all relationships between migration programs are covered by the used data subset. To give a counterexample: the partial migration run test must not consist only of customers without safekeeping accounts, but also contain at least one customer with a safekeeping to cover a potential dependency between customers and safekeeping accounts.

The *interference risk* is an infrastructure risk, which cannot be addressed by testing. It must be coped with on the organizational level. The *parameterization risk* requires to determine whether all data could be migrated, whether some data was refused, or if the data was migrated but the target application crashes. Thus, mitigating the parameterization risk can be covered by the combination of testing techniques used for address the completeness, the semantics, and the corruption risk. Table I provides a summary by linking risks and the suitable testing techniques.

### C. Complement the Process Model with Testing Phases

When it is clear which test to perform, the right execution order has to be worked out. In software engineering, the testing sequence is clear: unit testing, integration testing, and user acceptance testing. Only if the basics work, testing moves on to the more complex challenges. The same principle applies to testing in data migration projects. It is fundamental that the migration run test succeeds first. If some migration programs are not invoked, crash, or are executed in the wrong order (e.g., equity positions of safekeeping accounts are migrated before the safekeeping accounts) the data might be corrupt due to these errors. Only if this is clarified, step two can be tackled, hence appearance and completeness & type correspondence. This is followed by the processability test, which is succeeded by the integration tests in case of success. However, the ordering of the tests is not simply one-dimensional, the business object types form a second dimension. When the customers are tested correctly regarding appearance and completeness, the processability tests for customers can be initiated even if the safekeeping accounts might fail the appearance test. Making such decisions requires to know the dependencies between the different business object types. The data migration process

TABLE I  
TESTING-BASED RISK MITIGATION

Risk Group	Risk	Risk Mitigation
Subprogram	Stability	– Full migration run test – Partial migration run test
	Corruption	– Appearance test – Processability test – Integration test
	Semantics	– Appearance test – Processability test – Integration test
	Completeness	– Completeness & type correspondence test
Data migration program risks	Execution risk	– Full migration run test
	Orchestration risk	– Full migration run test – Partial migration run test
Infrastructure	Dimensioning	– Full migration run test – Partial migration run test
	Interference	<i>Operational risk, no testing</i>
Target application	Parameterization	– Appearance test – Processability test – Integration test – Completeness & type correspondence test

model (Figure 2) reflects the ordering of the five test phases (step 7-11).

The final rehearsal (step 12) covers all tests from steps 7-11, but in a condensed form. The final migration (and thereby do a final test) is practiced in the final rehearsal step by iterating through all phases of the process model [3], [13], [28]. Thereby, it has to be ensured that scope, approach, as well as boundary conditions conform to those of the productive data migration.

The same tests as done during the final rehearsal are re-executed in the productive migration (step 13). When all data has been migrated into the target application, the project team re-conducts all tests: appearance testing, processability testing, integration testing, completeness & type correspondence testing. Similar to the full migration program run test, the logs are checked for additional problems. When all tests succeed, the company decommissions the source application and switches to the target application for their daily business.

## VII. PROJECT MANAGEMENT-BASED QUALITY ASSURANCE

This section describes the organizational techniques for quality assurance in data migration projects. Presented findings originate from our insights when participating in data migration projects and are underpinned by statements made by relevant literature.

### A. Involve an external data migration team

Given that the company internal IT staff is often not familiar with data migration projects while behaving reluctantly, over-run of time or budget, in a worst case a complete project abort, are not unusual [1], [29]. An external migration specialist addresses the IT management risks of extended delays and overspends by bringing in practice-proven methodologies and tool support paired with long-lasting experience and in-depth know-how.

### B. Exercise due while perform project scoping

A comprehensive determination of the data being part of the migration endeavor during the strategy and pre-analysis phase is crucial to prevent the risk of data and transformation loss. The two diametral principles source-push and target-pull (cf. [3], [11]), thus migrate what the source offers vs. migrate what the target requires, ensure that all business objects in scope are completely migrated to the target database ruling out the risk of completeness.

### C. Apply a data migration platform

Using a well-tested data migration platform (instead of developing a new one or none) has several advantages: independence from source and target applications' particularities, a dedicated single point of truth, or the reuse in future migration projects [10], [13], [22], [25]. In particular, the fact that the platform provides increased migration leeway for the data migration team, i.e. allowing bogus transformation and preliminary tests, helps to mitigate the risks of corruption and instability which is higher when migrating directly from source to target. Since the platform is scalable, the migration speed can be adjusted afterwards [13]. This mitigates the dimensioning risk of having a large amount of data handled by an undersized infrastructure. From an IT management risk perspective, the migration platform enables a high parallelization of work right from the start, given that the data migration team and target application parameterization team can work independently from each other [10], [22]. Performing migration and parameterization simultaneously prevents budget and time overruns while reducing the risk of interference between the teams. Except for the initial extraction and final loading of the data, interference with source and target business application including its data is completely avoided [25]. Thus, the parameterization risk and the impact a change in the target application might have on the migration project is reduced.

### D. Thoroughly analyze and cleanse data

Investigating the data in detail contributes to the understanding of its semantics and structure. The data migration team can evaluate inconsistencies, redundancies, inaccuracies, and referential integrity across tables and attributes [16], reducing for example the corruption risk. Furthermore, costly data mapping rework is reduced preventing also project delays [16]. As a by-product of the analysis, the project's characteristics can be seized more accurate addressing the IT management risks of project delays and budget overruns. An subsequent cleansing makes sure, that the target application is not polluted with redundant or even dirty data possibly influencing the application's performance and stability. Moreover, clean data brings down the risk of unstable data migration programs.

### E. Migrate in an incremental and iterative manner

From an IT management perspective, an incremental, step-wise approach reduces the risk of a project failure. Results are generated early and often ensuring a high traceability and the possibility for frequent adjustments. In particular,

the migration to databases of well-known standard software products allow for code re-use, hence the re-employment of previously developed transformation programs or parts of them. These programs encapsulate acquired knowledge with regards to past transformation problems while having been thoroughly validated during comprehensive test runs.

Combining data migration specific experience, general best practices from the ever growing-body of literature in project management and software development, as well as testing techniques of the previous section helps to assure a high quality and mitigate the risks typically arising in data migration projects.

## VIII. CONCLUSION

According to the Gartner Group “83% of data migrations fail outright or exceed their allotted budgets and implementation schedules”. To deliver a data migration project in time and on budget, a combination of a stringent approach, proactive risk mitigation techniques, as well as comprehensive test activities are indispensable. This paper addresses the rarely investigated field of quality assurance and testing in data migration projects and therefore contributes to the topic of maintenance. Based on a practice-proven process model describing how the authors of this paper proceeded when shifting data from a source to a target database, different risks are identified, described, and exemplified. Aiming at eliminating these risks, several best-practice mitigation techniques are introduced and classified before linking them to the different phases of the process model afterwards. Finally, dedicated project management practices also helping to overcome risks are presented.

Business events like M&A where databases with almost identical business object types (e.g. customer vs. client) have to be consolidated require a harmonization phase. However, data harmonization has not been addressed by our process model and should be considered in future studies. The situation, where the content of more than one database has to be migrated to the target also falls out of the article’s scope. Likewise to harmonization, this scenario brings along new risks which have to be addressed by appropriate techniques. Even if the presented process model and risk mitigation techniques originate from our empirical observations in practice, future research should systematically evaluate them through empirical studies. In doing so, the probability for the occurrence of a risk as well as the success rate of a certain mitigation technique can be determined. Lastly, further work should center around alternative versions of the model and the techniques applicable for non-relational databases.

## REFERENCES

- [1] P. Howard and C. Potter, “Data migration in the global 2000 - research, forecasts and survey results,” London, United Kingdom, p. 29, 2007.
- [2] F. Matthes and C. Schulz, “Towards an integrated data migration process model - State of the art & literature overview,” Technische Universität München, Garching bei München, Germany, Tech. Rep., 2011.
- [3] Endava, “Data Migration - The Endava Approach,” London, United Kingdom, p. 11, 2007.

- [4] A. Freitag, F. Matthes, and C. Schulz, “M & A driven IT transformation – Empirical findings from a series of expert interviews in the German banking industry,” in *10th International Conference on Wirtschaftsinformatik (WI2011)*, Zürich, Switzerland, 2011.
- [5] G. Schröder, “Automatisierte Migration von Legacy Daten,” Diplomarbeit, Fakultät für Informatik, Technische Universität München, Garching bei München, Germany, 2006.
- [6] C. Burry and D. Mancusi, “How to plan for data migration,” 2004.
- [7] K. Dongre, “Data Conversion Process Design Methodology for OLTP Systems,” Brookfield, WI, USA, 2004.
- [8] IBM, “Best practices for data migration - Methodologies for assessing, planning, moving and validating data migration,” Somers, NY, USA, p. 16, 2009.
- [9] K. Haller, “Data migration project management and standard software – experiences in avaloq implementation projects,” in *Data Warehousing Conference - DW2008: Synergien durch Integration und Informationslogistik*, St. Gallen, Switzerland, 2008, pp. 391–406.
- [10] W. Lashley, “Legacy Data Migration: DIY Might Leave You DOA,” Seweckley, PA, USA, p. 8, 2006.
- [11] K. Haller, “Towards the industrialization of data migration: Concepts and patterns for standard software implementation projects,” in *21st International Conference on Advanced Information Systems Engineering (CAISE)*, Amsterdam, Netherlands, 2009, pp. 63–78.
- [12] M. Willinger and J. Gradl, *Datenmigration in SAP*, 2nd ed. Bonn, Germany: SAP Press, 2007.
- [13] G. May, “Datenmigration,” München, Germany, p. 35, 2010.
- [14] P. Russom, “Best Practices in Data Migration,” Renton, WA, US, 2006.
- [15] J. B. Shepherd, “Data Migration Strategies,” Brookfield, WI, USA.
- [16] Informatica, “The five pitfalls of data migration - and how to avoid them,” Redwood City, CA, United States, p. 16, 2010.
- [17] C.-Y. Lin, “Migrating to Relational Systems: Problems, Methods, and Strategies,” *Contemporary Management Research*, vol. 4, no. 4, pp. 369–380, 2008.
- [18] J. Morris, *Practical Data Migration*, 3rd ed. Swindon, United Kingdom: British Informatics Society Ltd, 2006.
- [19] B. Wu, D. Lawless, J. Bisbal, R. Richardson, J. Grimson, V. Wade, and D. OSullivan, “The Butterfly Methodology : A Gateway-free Approach for Migrating Legacy Information Systems,” in *3rd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS97)*. Como, Italy: Institute of Electrical and Electronics Engineers, 1997, pp. 200–205.
- [20] V. Anavi-Chaput, K. Arrell, J. Baisden, R. Corrhons, D. Fallon, L. Siegmund, and N. Sokolof, “Planning for a Migration of PeopleSoft 7.5 from Oracle/UNIX to DB2 for OS/390,” Poughkeepsie, NY, USA, p. 148, 2000.
- [21] P. Manek, “Microsoft CRM Data Migration Framework,” p. 24, 2003.
- [22] D. Aebi, “Re-Engineering und Migration betrieblicher Nutzdaten,” Ph.D. dissertation, Eidgenössischen Technischen Hochschule Zürich, 1996.
- [23] DataFlux, “The three key phases for data migration - and beyond,” New York, NY, USA, 2009.
- [24] M. L. Brodie and M. Stonebraker, *Migrating Legacy Systems*, 1st ed. San Francisco, CA, US: Morgan Kaufmann Publishers In, 1995.
- [25] G. Plivna, “Data migration from old to new application: an experience,” 1997.
- [26] K. Haller, “Datenmigration bei standardsoftware-einführungsprojekten,” *Datenbank-Spektrum*, vol. 8, no. 25, pp. 39–46, 2008.
- [27] —, “On the implementation and correctness of information system upgrades,” in *26th IEEE International Conference on Software Maintenance (ICSM)*, Timisoara, Romania, 2010, pp. 1–5.
- [28] D. Schlier, R. Karcher, and C. Fischer, *Migrationsprojekte von Kunden der Finanz Informatik zur Gesamtbankenlösung OSPlus*, 1st ed., St. Gallen, Switzerland, 2009, ch. 4, pp. 134–144.
- [29] P. Allaire, J. Augat, J. Jose, and D. Merrill, “Reducing Costs and Risks for Data Migrations,” Santa Clara, CA, USA, p. 31, 2010.