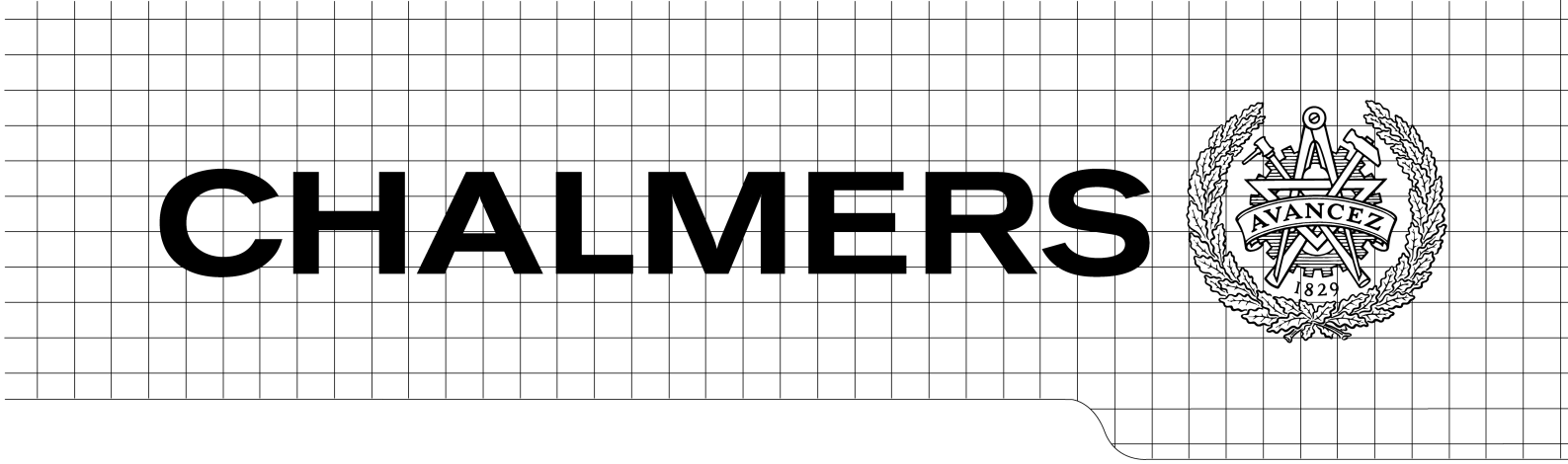


Data migration, a practical example from the business world

Master of Science Thesis in Software Engineering and Technology

LINA RINTAMÄKI

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, August 2010

Data migration, a practical example from the business world

L. RINTAMÄKI

Examiner: P. ZARING

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden August 2010

Abstract

A high amount of data is being managed by databases and applications in companies today. In many cases data is extracted from several different databases, then managed and finally stored in yet another database. The process of moving data from one database to another is called data migration. The data migration process can eventuate in several new problems, especially considering the high amounts of data that is being processed.

In some cases the data migrations are conducted manually, which can be very time consuming. On the software market today there exists several different 'off the shelf' products that manages the data migration process. The problem is to find the one that is most suitable for the company, or to realize that there is none that is suitable enough.

The practical work performed behind this masters thesis is the development of an application for data migration that is custom-made for the company Dipcon and their specific needs. In addition to the data migration the application also performs the transformation of the data that is needed before the migration.

The more theoretical part of this thesis is focused on the theories and practices behind data migration. Data migration projects are very common, a clear majority of the companies asked in Bloor Reasearch's *Data migration in the global* are performing a data migration each every 3d month or more often. Even though they are so common, the data migration projects seem to be misunderstood with a more than 60% failure rate. This thesis is discussing the reasons behind this and have some suggestions to add in order to increase the likelihood of a successful data migration.

Acknowledgements

I would like to thank Gustav Sanders at House of Ports for the opportunity of developing there new application for the yearly invoice system, as well as for the support and guidance during the project.

Table of contents

1. Introduction

This is a masters thesis conducted at the masters programme Software Engineering and Technology at Chalmers University of Technology in cooperation with the company House of Ports.

It concerns a practical example of a data migration project between two databases, as well as the theory and practice behind data migration. The practical example consists of a display and review of an assignment from the company House of Ports and their affiliate Dipcon. The assignment was to develop an application that can manage the yearly invoices of Dipcon and involves a data migration of the invoices.

1.1 Background

A majority, if not all, companies today uses a database system of some sort and manages a high amount of data. In some cases the data has to be transferred to another database system, that process of transferring data is called data migration. Since it takes an incredible amount of time to transfer the data manually, several software products has been developed that manages different data migrations. However, it can be hard to find a suitable off the shelf product so sometimes a data migration application has to be custom made and developed in house. The application developed for Dipcon is of the latter.

1.1.1 House of Ports & Dipcon

This thesis has been developed in a close cooperation with the Swedish company House of Ports.

House of Ports is a company that combines five different business areas and offers products and services within domain names, trademarks, jurisprudence, consulting, design and hosting. The affiliated company of the concern House of Ports that will be using the resulting application of this work is called Dipcon, domain and intellectual property consultants.

Dipcons business area is to assist other companies with the administration of domain names and trademarks and also consultant services like dispute resolutions, strategies and trademark registrations. Each year an yearly invoice is sent out to all of their customers.

The system they have been using up until now is not well suited for the increased amount of data that has come with the expansion of the company. This has resulted in a need for a lot of manual work when the yearly invoices have had to been sent out. Therefore a new system for managing the yearly invoices is required.

1.2 Problem statement

The problem discussed in this thesis is how to increase the likelihood of a successful data migration. A hands on approach has been used, where a data migration project has

been conducted. In order to provide a solution to the problem, general theory and practice behind data migration is discussed as well.

In addition to the data migration, the developed application also manages the computations of the data that is required for the calculations of Dipcons yearly invoices.

1.3 Purpose and goal

The purpose of this thesis is to provide a wider understanding of the challenges facesd during a data migration between different systems through a review of a practical example and discussion about the difficulties and risks involved.

The specific goal is to develop an application that manages Dipcons yearly invoice system.

1.4 Delimitations

There is several different types of data migration, the data migration process is simply the process of moving data between different storage units, computer systems or formats. This thesis is focusing upon a practical example of data migration between database systems and the work that is needed to make a migration possible.

The theoretical part about data migration however, is not limited to strictly database migrations, but includes all kinds of data migrations and the common challenges.

Due to the privacy statements of House of Ports, the appendix with the class and sequence diagrams is not included in this thesis.

1.5 Definitions and abbreviations

ccTLD	(Country Code TLD) A TLD used or reserved for a country.
Database	A collection of data stored on a computer and organized in such way that it is easy for a computer program to find a desired piece of data.
Data migration	A transfer of data between storage units, computer systems or different formats.
DBMS	(Database Management System) A set of computer programs that manage the creation, maintenance and use of a database.
Dca2	A postgresql 8.3 database at House of Ports.
DMS fee	(Domain name Management fee) The maintenance fee for the yearly invoice of Dipcon.
DNS	(Domain Name System) The DNS is a naming

	system for computers, services or any resource connected to the Internet. It translates human-readable names into binary addresses.
Domain(database)	A MSSQL 2008 database at House of Ports.
Eclipse	An open source software development environment.
Eclipselink	A software that provides an extensible framework that allows Java developers to interact with various data services, including databases, web services, Object XML mapping (OXM)
ETL	(Extract-Transform-Load) A process that changes the uses of data. Usually from an operational environment to an analytical.
Extract	The process of reading data from a database.
gTLD	(Generic TLD) TLDs like .com .net .org.
GUI	(Graphical User Interface) A user interface that offers graphical icons and visual indicators as opposed to a text-based interface.
HOST fee	(Hosting fee) The fee for the different hosting services a customer is using in the yearly invoices of Dipcon.
IDN	(Internationalized Domain Name) A domain name that contains at least one label that is displayed in software applications in a language-specific script of alphabet, such as Chinese, Russian or Swedish (ääö).
JPA	(Java Persistence API) A Java programming language framework that allows developers to manage relational data in applications using Java Platform, Standard Edition and Java Platform, Enterprise Edition.
JPQL	(Java Persistence Query Language) An object-oriented query language defined as part of JPA.
Load	The process of writing data to a database.
ORM	(Object-Relational Mapping) A programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages.
PNMS	A MSSQL 2000 database at House of Ports.
Portek	A MSSQL 2000 database, the economy database at

	House of Ports.
REN fee	(Renewal fee) The fee for the renewal of domain names in the yearly invoices of Dipcon.
SQL	(Structured Query Language) Standardized query language for requesting data from a database.
TestNG	TestNG is a testing framework for the Java Programming Language.
TLD	(Top Level Domain) A domain at the highest level in the DNS. For all domains in lower levels, it is the last part of the domain name. For example, in the domain name www.example.se , the top-level domain is 'se'.

1.6 What is data migration?

This thesis is about data migration, both the theory and principles behind a data migration and a practical example from the business world. For that reason a short introduction to data migration may be required before the review of the data migration project. More thorough information about data migration theory and practice is available in section 4 the purpose of this section is only to provide enough information to make the review easier to understand from a data migration point of view.

The process of moving data between different databases, computer systems or storage types is called data migration. The data migration project conducted at House of Ports concerns a transfer of data between different database systems.

The data migration process can be performed manually, but it is very time consuming to do that, so most often it is automated, through different software products or by hiring an data migration company. In order to achieve an effective data migration, a transformation of the data on the old system may be required in order for the data to conform with the new system design. This is often done by developing a design that matches the old data to the new database's format and requirements.

Data migration projects can include several steps, but at least two steps are required, data extraction, where data is read from the source database and data loading, where data is stored on the destination database. In the practical example discussed in this thesis a third step is involved, placed in between the extraction and the loading. That step manages the transformation of the old data so that it matches the new database's format. When the data migration is completed, a fourth verification and validation step is required to make sure that the data was accurately translated,is complete and that there is no loss of data. The verification process often involves a parallel run of both the old system and the new one.

2. Method

The assignment from House of Ports and Dipcon was analyzed together with the supervisor at House of Ports, Gustav Sander, and then divided into sub problems, or steps, to be developed in order. Along the way, functionality and requirements have both been added and subtracted from the main assignment, but the key part has been kept intact.

Each step began with a discussion with the supervisor and a discussion with the staff at Dipcon, the end users of the system. After that, some time was spent to get to know the database systems, to profile the data, to get a clearer picture of what had to be done and where the required data was located. When the information about the systems was gathered, some new questions usually had appeared and it was time for some more discussion with the business people to clarify them. The gathering of information about the database systems was most necessary and time consuming in the beginning of the project, later on the required information was already acquired.
Since this project is both about the practical example of a data migration at House of Ports and about the theories and practices around data migration, the project is rather in two more or less independent parts.

The method for the second part, the principles behind data migration, is much more theoretical than the practical example, so it consists mostly of literature studies. The overall approach to the subject has been a version of “trial and error” with the data migration project being conducted before the literature study for theory and practices about data migration. That supplied with a hands on knowledge about some of the most common errors during this kind of project, and a real understanding of the necessity of some of the theories and practices. It is, however, not a method to recommend for someone that is planning on conducting a data migration.

2.1 Literature study

The theory behind data migration is what dominated the literature study, but some literature study was needed for the data migration project as well. However, the literature required for the development of the application, which is more oriented towards common programming challenges is not included in the references. Only the literature that contains information about the subject data migration is included.

A majority of the literature has been accumulated on the internet, because the area is quite new and it was difficult to find any information in books at the libraries.
The literature study for the practical work has been conducted in parallel with the work when it was required.

The majority of the literature study for the theory behind data migration, however, was scheduled to be conducted later on, when I had gotten a grasp about it from the practical example. Initially only a first introduction to the area seemed suitable. Later on more information about the data migration was required and thus the literature study for this

part was conducted during the mid phase of the project.

2.2 Work method

The majority of the work focuses around the practical example of a data migration and the transformation of the data that has to be done in order to migrate it.

The application has been developed in house at the customer, House of Ports in Jonsered, Göteborg. This provided a close contact with the intended customer and with the projects supervisor at the company.

The assignment was to develop an application that manages the yearly invoices of Dipcon and it has been developed in three independent steps, the computations to create the debts, the migration of the debts to the economy database, including transformations of data, and the GUI for the computation step.

The steps were implemented in order of significance. Since the computation and transformation part and the migration part was the most important ones, they were scheduled first and the least important step, the GUI, was scheduled in the end. Due to lack of time at the end of the project, there was not time enough to complete the GUI before the presentation of this thesis.

3. Theoretical background

Data migration is the process of transferring data from one system to another. In the case of the reviewed data migration project, the systems were four databases.
Data migration differs from data movement, data integration and conventional ETL (extract, transform and load) respectively by migrating to a new environment, by being a one-time process and by maintaining the consistency of usage.[2]

The success rate of data migration projects is low, although not as low as in other software projects. Since the data migration process in most cases manages a lot of valuable data, as in the case of a data migration for a bank, a migration can be very risky process. That may be the reason why many companies postpone the required migration for as long as possible. However, that approach leads to added difficulties when the migration no longer is avoidable. That is due to that older hardware and software may require more manual work and maintenance during a migration and that they generally has lower performance and is more prone to failure[3].

The time it takes to perform a data migration if one only considers the extract and load phases is a function of the size of the data objects, the speed in the transferring medium and the degree of parallelism. From an algorithmic point of view the data migration problem is the problem of computing the most efficient plan for moving data from a source to a target[5]. The migration algorithm can be reduced to the edge-coloring problem, and thus is a NP complete algorithm. However, this thesis won't discuss the algorithms in more detail, that is a thesis in it's own.

Since the data migration project concerned in this thesis is a data migration between different databases and the data migrations concerns an invoice system for a company which business area is to provide services for domain names, this section starts with the theoretical background behind databases and domain names.

3.1 What is a database?

The data migration involved in the development of the application during this thesis is a migration between database systems. In order to get a full understanding of the migration and the developed application, this section will give a general introduction about databases and database systems.

With the computers and the more complex applications came a need for data that is persistent even after an application has been shut down. Databases solves that problem and can be thought of as a collection of information that exists during a long period of time. The collection is managed by a Database Management System, DBMS.
The DBMS provides the users with:

- The ability to create databases and decide how the data will be structured in the database.
- The ability to query a database for the wanted data, and to modify that data.

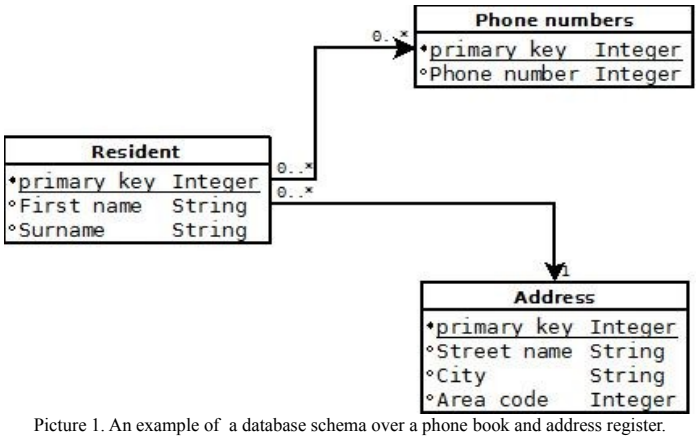
- The ability to store very large amounts of data over a very long period of time.
- The ability to control the access to data from many users at the same time, without any “collisions”, for example if one user modifies data while another user is reading it.

The first database systems appeared in the late 1960s where they began to replace the previously used system of storing data in the file systems. The advantage of using databases over the file system was partly the superior access to the data. It's hard, if not impossible, to know where in a file a certain piece of data is stored, while in a database it's easy to construct a query to find the desired data[1].

The first important applications of database systems were systems like banking systems, airline reservation systems and corporate record keeping. Common for these is that the data consists of many small parts and that the data is often read and modified several times.

There exists several different DBMS, the most popular one today is the relational database management system. It was proposed in the 1970s by T. Codd[1] and is the one used by House of Ports and thus the most interesting one in this report. A relational DBMS is a DBMS in which the data is stored in the form of tables and the relationship among the data is also stored in the form of tables.

One may think of a DBMS as a phone directory, and a query can select a sub set of information of the data in the phone directory. For example, it can select the phone number to everyone with a specific surname or a specific first name. While it is easy to find a phone number in the phone directory if one knows the persons surname and first name, it's almost impossible or at least very time consuming, to find the name of a person if one only knows the phone number. In a database, both of those things are as simple to find out.



Picture 1. An example of a database schema over a phone book and address register.

In a relational database with a table consisting of the residents and a table with phone numbers, and each resident has a phone number that is stored in a table consisting of phone numbers, then the resident has a *relation* to the phone number.

One resident may have zero or more phone numbers, and each phone number can belong to many residents (if they live together or share the phone). More over, each resident may have an address and if so, then there is a relation between the table of residents and the table of addresses as well.

To identify a certain row in a table a unique value, a primary key, is used. For example, the table of residents may have the residents social security number as primary key, or a primary key that is independent of the data in the table and just used to identify the specific rows.

The relations between tables is represented with foreign keys. Each resident has an address, so in the resident table there is a column containing the addresses primary key, in that case, the primary key is referenced as a foreign key.

3.2 What is a domain name?

Since Dipcons business area in large amount concerns managing domain names, a short introduction about domain names is needed in order to gain a better understanding of the review of the application.

A domain name can be thought of as an identification label based on the Domain Name System (DNS). The domain names are ordered in hierarchical layers of the DNS root domain, which has no name. The highest level group of domain names are called top-level domains, TLDs, a group composed of the generic top-level domains, gTLDs, the country code top-level domains, ccTLDs and a group of infrastructure TLDs, containing TLDs like ARPA. Below the TLDs in the hierarchy are the second-level and third-level domain names. The purpose of the domain names are to provide human readable names as opposed to the numerically addressed internet resources. Domain names are often referred to simply as domains, so also in this thesis.

A domain name consists of one or more parts, labels, that are delimited by dots. The right-most label of the domain is the top-level domain. For example www.gothenburg.se belongs to the (country code) top-level domain se. The hierarchy of domains descends from the right to the left label in the name, each label to the left is a sub domain of the domain to the right. Each label may contain up to 63 ASCII characters and the full domain name may contain up to 253 characters.

The character set allowed in the DNS initially prevented the representation of names and words of many languages in their native scripts or alphabets. ICANN later approved the Punycode-based Internationalized Domain Name, IDN, system, which maps unicode strings into the valid DNS character set. For example, www.göteborg.se is mapped to www.xn—gteborg-90a.se.

3.3 Statistics and data migration success rates

In *Softeks 2005 Worldwide Data Migration Survey* 60% of the respondents answered that they performed a data migration once every 3^{d} month or more often. As many as 19% answered that they performed data migrations on a weekly basis. More than 75% of the respondents answered that they had experienced some kind of difficulties during the migration.

Other research[2] has concluded that as many as 64% of the data migration projects fail, either by going over time or going over budget or, as in many cases, both time and budget. The main reason for the overruns in time and budgets appear to be the difficulties to successfully plan and scope the project. The data migration success rates may seem like very poor results and may be an indication on that a more standardized methodology for data migration is needed. However, when comparing to other software projects, the findings are not so deviant. With an average of only around 16% of the software projects being defined as successful[9], that is, they are delivered on time, in budget and with the expected functionalities, the data migration projects actually shows a higher delivery rate.

While a majority of the data migration projects ran over both time and budget, the main reason they ran over budget was because of the time overruns. And it's easy to understand that a time overrun in most cases, if not all, results in an overrun in the budget.

According to the answers to the Bloor research[2], the second most influencing reason for the overruns in time and budget, besides the unrealistic scoping, is the lack of methodology. That implicates that there exists a need for a more standardized data migration methodology in order to achieve a more successful data migration.

3.4 Why do things go so badly wrong so often?

As the statistics shows, things often go badly wrong during a data migration project. Even if all software projects have their flaws, there is certainly room for improvement in the general execution of a data migration project. Some factors seems to have more importance than other. According to the study performed by Bloor Research [2], the one key issue is the scoping of the project. While there is only a small percentage that actually refers to the scoping as the problem, several more refers to issues that can be interpreted as some kind of scoping issue. With those in account, a majority of the projects may have failed to be completed in time and budget due to scoping issues. In order to improve the result from a data migration project, the scoping appears to be the key to work with.

The reason why it is so hard to scope a data migration project may be that even though a majority of the projects claims that they are using a formal methodology, it may be hard to follow them and understand why each step in the methodology is important. An important thing to notice is that while the answers to the question about what went

wrong pointed to scoping issues, a clear majority of 85% also answered that their scoping activities were successful. This suggests that there is some kind of a misunderstanding about the scoping. One reason may be that a data migration is mistaken for a “big-bang” event instead of a process in motion[10]. Some scoping and planning is performed and then the migration is supposed to be completed as soon as possible. However, a migration is more often a long process with several passes that should be performed. This view of the data migration project may be due to that the data migration often is a part of a larger project and is therefore viewed as a step in that process and not a project of it's own[12].

Another reason why the scoping often is inadequate is that teams that has performed migrations before scopes the project at hand according to how the last migration went, but no two migrations are the same and more time and effort is needed for the profiling of the data in order to fully understand the new project[11].

The other key issue behind delays in data migration projects are unexpected problems that occur during the course of the migration[2]. Unexpected problems range all the way from the entire project team going down with flu, to problems that occur due to lack of an understanding of the problem at hand. However, the kind of unexpected problems that belongs to the other end of the spectrum shouldn't be unexpected at all and, in fact, is due to inadequate scoping.

3.5 A data migration methodology

When implementing a data migration methodology the most important step is to document the possible risks and the likelihood that they will occur and to limit the likelihood or else mitigate the risks. This is due to that all data migrations are different and risks are specific for the project at hand, so no methodology can cover a full risk estimation.

The data methodology proposed here is overall the one proposed by IBM[3], with the three main steps; plan; migrate; and validate. There is however, some changes from their methodology, some steps have been added, and others removed. There is an emphasis on the steps that covers data profiling and the scoping of the project, since they are identified as the steps to work with in order to achieve a higher likelihood of success.

3.5.1 Agility in the migration

The proposed methodology may seem to be a waterfall process; take this step first and take that step after that. But due to the complexity of the data and the source and target applications there is a need for agility and flexibility during the data migration. A data migration project without adaptivity is likely to discover the data problems late in the project, when they are most expensive and likely to cause the longest delays[12]. Therefore it is necessary to be open to iterative and flexible development with respect to the data and never assume that the project team has all the required knowledge and

information about the data.

3.5.2Split the migration into two or more parts

A data migration is a complex transaction involving several different changes that has to be done to the data. In his article *Database migration – A planned approach* [7], Steve Callan suggests that in addition to the planning, migrating and validating phases there may also be an idea to split the migration into two or more parts. Each part has it's own inherent risks and if the migration of one part should go wrong, why let the other parts follow. It might seem more effective to take it one part at a time and make sure that that part works before adding the next part to the migration. However, the added costs behind each split that is used is not noted by Callan.

Others, like Gershon Pick [13], is opposed to the view presented by Callan. They argue that the migration process is a delicate business, and it is better to assure progress through extensive planning and testing then by using a method that reminds more about trial and error. And for each used split more complexity is added to the migration in order to maintain the parts and with added complexity comes added risk.

With that in mind, the technique with splits were used during the migration described earlier in this report. One split was used, and it was used between the transformation and the migration steps. The decision was made partly because the migration step without the transformation step could be used for more migrations then just this one, and partly to open up for more verification and validation of the transformed data before the migration step. Since the added risk from the new complexity is risk mostly involving the time factor and there was more time available for the project, the decision was not that hard to make.

3.5.3 Planning phase

The planning phase in a data migration project is often reduced and something just to get over with, but it is the most important step when wanting to deliver a successful migration in time and on budget. The planning phase may take longer time than first expected, a couple of days, or weeks depending on the size of the project, can be a realistic time scale.

The planning phase for the practical example described in this thesis consisted primarily of data profiling and to get the business knowledge. Since the communication during the development of the practical example in this thesis was with a programmer that already had an understanding of the importance of data scoping, the task of scoping became much simpler and consisted mainly of data profiling. No off-the-shelf products were used since the project was developed at the company and House of Ports, the company, already had availability to the required products needed for data modelling etc.

3.5.3.1 Scoping

As discussed earlier in this thesis, the main reason behind the extensive failure rate of data migration projects is the lack of accurate scoping. The reasons why it is so difficult to perform an accurate scoping is described in section 4.2 and will not be discussed in detail in this section. This section is focusing on what a project team can do to increase the accuracy of their scoping.

One important step to take is to make sure that the client understands the importance of the scoping and the risks that may occur due to the lack of it. Without understanding the benefits, it's easy for the client to view the scoping as time consuming and unnecessary.
In his article *The problem with data migration scoping,* John Platten argues that the most important steps to take to avoid the problem with scoping is to:

- Profile the data through all the maps and schemas needed.
- Communicate the importance of the scoping to the client.
- Investigate the business consequences of a failure.
- Have a conversation about funding the scoping phase.

This can be summarized as two main steps, make sure to thoroughly profile the data and learn how to communicate the necessity of scoping to the client. More in depth tips about data profiling follows in the next section.

3.5.3.2 Profiling the data using landscape analysis

Profiling of the data is a necessary step in order to ensure a successful project. Project estimates that is done before data profiling are often more based on guesswork or previous experiences with data migration projects, than on an accurate assessment of the data migration project at hand.[15]

The project team needs to know if the required data exists, where it is, how it is related, eventual standardizations, if the data is detailed enough, if there is data that is unstructured or orphaned and what the focus is of each source[12].
Although there is many ways to achieve the required understanding of the data, this thesis will focus on landscape analysis which for instance involves logical and conceptual models.

Landscape analysis is a way of performing a data migration simulation and includes 4 activities:

- Business process analysis, which focuses on the differences between the source and the target system from a business point of view, this is done through a collaboration between technical- and business staff.
- Data modelling analysis that make use of the pareto rule and focuses on the 20% most critical data, typically the primary keys. The data modelling starts at the

top, with conceptual models, and continues with logical and physical models in order to get deeper and deeper into the details.
- Data discovery analysis that is using the models developed during the data modelling analysis in order to determine what technologies and tools to use. Both technical- and business staff should be involved in the decisions.
- Mapping analysis is where it all come together and where the big issues are found and measured. Mapping analysis is about identifying the possible challenges and risks during the migration from the source system to the target system.[11]

The landscape analysis is a quick way to get to know the source data, the target system and to discover possible threats and challenges to overcome. Since time often is of the essence, usage of the pareto (20/80) analysis is beneficiary.

During the project at House of Ports the data profiling was made during the course of the project and in close contact with the business people and with the aid of conceptual and logical models. It took quite long time to achieve an adequate understanding of the source data and the target system, that time could possibly have been decreased by the use of more extensive data profiling at the start, like the landscape analysis.

3.5.3.3 Business knowledge

One broad misconception about data migration projects is that they only concern the programmers and the technical staff. However, the criteria that constitutes a successful migration is not, at least should not be, technical but business-oriented[10]. If the criteria was purely technical, projects with data that has been migrated from the source system to the target system without problems would be viewed as a success. However, it's critical that the migrated data is formatted and can be used by the target system and the users of the system. When end users is brought into the project first in the last steps, in order to test the new system with live data, they are likely to find fundamental problems that are very time consuming and expensive to fix late in the projects life-cycle.[13] Due to that, a close contact with the business people is required during the course of the project.

During the development of the application described in this thesis there was a close contact with the intended users. Since the application was developed at the company House of Ports, the intended users were at the working place full time and were ready to spend time on answering questions and to check things out.

3.5.3.4 Consequences of a failure

Since the statistics are as they are, with a clear majority of data migration projects failing, there are benefits to gain from the knowledge of the consequences of a failure. Especially when considering off the shelf products and outsourcing. In many cases the consequences of a failure is far more expensive then the most expensive products and data migration companies.[13]

3.5.3.5 Selecting off the shelf data migration product

Data migration is a process that is done several times and by most, if not all, companies. Therefore there exists several off the shelf data migration products that are specialized on different types of migration. While it can be difficult and time consuming to find the right product, there might be time and money to save. In addition to the developed products, there may exist available data migration companies that can develop a custom made solution. The key factor to bear in mind is the consequences of a failure and that the costs of an overrun in time and budget can become very expensive and completely dominate the expected net cost of the project.

According to IBMs *Best practices for data migration* there is five primary factors to consider when choosing data migration software[3]:

- Performance, how quickly the data can be migrated
- Primary volume/source data protection, if something goes wrong, the migration can be terminated and restarted or continued later without any loss of data
- Tiered storage , moving data between different storage media without disruption
- Multivendor environment, that the software allows for the source system and target system to be from different vendors
- Application downtime, how long time the system will be down during the migration

Since each data migration project is unique, even if they have a lot in common, it is difficult to advice on specific data migration software. The choice of using off the shelf products should be taken after the data profiling phase, when as much as possible is known about the data and the source and target systems.

3.5.4 Migration phase

The migration phase of the data migration involves more than just export and import phases. Possible transformation of the data is performed during this step and according to some methodologies some data modelling is done is the migration phase. However the recommended methodology in this thesis groups all the data profiling and data modelling in the planning phase of the migration, and the migration phase focuses on export, transformation and import.

Before the export phase, but still during the migration phase, is it recommended to perform a pre migration data validation test.[3] This test can later be matched with a post migration data validation test to ensure that the data is in the same state after the migration as before the migration.

3.5.4.1 Export phase

During the export phase data is exported from the source system. If the time isn't of the essence, there is nothing that says that the export phase has to be run only once or in one bulk. Except that it requires more time, it might be an idea to consider splitting up the export into functional groups. If the export of data is broken up into groups, export and import of data may be run simultaneous, once one group of data is exported it can start being imported while the next group is getting exported. [8]

The export can be driven by an interactive mode, or by the use of shell scripts and parameter files. The key metrics when considering which way to maintaining the export is the integrity of the data and the available time.

When designing the export of data, it is necessary to decide what should happen if the export fails; should the export phase restart from the beginning or should it start from just before it failed. If this isn't considered and a failure occurs, data may be lost or there might be duplicates of some data.

3.5.4.2 Transformation phase

A need for a transformation of the data between the source and target systems are common. The main work done in the example application described earlier in this report is done in the transformation phase. Typically for the transformation phase is that it is handed by ad hoc programs that are coded outside of the other tools used for the migration if commercial off the shelf products has been used[7].

There is a critical need for the involvement of business people, as well as simple programmers, during the transformation phase, since the business people are the only ones who know how the data will be used after the transformation and the migration.

The data involved is often critical and in large amounts and in order to complete a successful migration every data record has to be transformed and migrated. Technical challenges that is likely to be faced during the transformation step of a data migration is synchronization and stabilization of the source data. The data to be transformed usually come from more than one source system, and the source systems may have different requirements and standards for the data. Data purification and error correction may also occur during the transformation step, an example from the application discussed earlier in this thesis is the control that a domain's name really is a correct domain name. [13] Value transformation is another challenge that occurred during the practical work of this thesis, the fees in different price list was in different format. In most cases they were integers and multiplied by 100 to give room for two decimals, but in some cases they were doubles and not multiplied by 100.

There is several technical challenges to face during the data transformation phase, but as seen in section 4.3.3.3, there exists some off the shelf products that may help with the transformations.

The approach taken in the practical example is to first ensure that all required data exists, and in some cases that the data is of the right format. If something is not correct, the application will ask the user to fix it before the record can be transformed and migrated. If all the required data for a record is correct, a record will be created out of that information, but it will still be stored on the source system. That is due to the need of more validation and verification testing and because there is an interest in all records being migrated at the same time.

3.5.4.3 Import phase

The import phase constitute of the load of data to the target system. At this point the data should be transformed to fit into the new system so all that is needed is the loading phase. As well as in the other phases of the migration step, a key issue is to know how to handle failures during the import phase in order to avoid data loss or duplications of data. All the import scripts or applications should be able to run several times without any data integrity lost.

3.5.5 Validation phase

Many data migration projects fails, and they do so mostly because of time and/or budget overruns. But there is a lack of statistics over how many target systems that work as intended immediately after the migration phase. A data migration project should always have a backup plan to fall back to if the target systems doesn't behave as expected once the migration is completed.

During the validation phase there is time for reviews of the migration done, and to compile statistics over what worked and what failed, in order to learn for future projects. The data migration is likely to be followed by more, and it is a good time to build a repeatable and consistent process for data migration[15].

If a pre migration data validation test was performed it may now be match with a post migration data validation test, to ensure that there is no differences between pre migration and the post migration data. But an independent validation of the migration is also necessary, one cannot trust the migration architecture to validate the migration.[15] This can be done by another project team at the company that get access to the requirements of the migration and of the new system and then develop a validation application, or by outsourcing the subject to a different supplier.

4. The application

The main part of this thesis is focusing on displaying and discussing the practical example of a data migration project and application that has been conducted for the company House of Ports.

The application is developed as two independent parts, since the company also have a separate use for the second of them. One part computes the yearly invoices and performs all the required transformations of the data, the other manages the migration of the invoices to the economy database and the post migration verification and validation. The data migration in this case consists of four steps, extraction of data, transformation of data, loading of data and verification of data. The first part of the application manages the first two steps, and the second part the third and fourth steps.

4.1 Introduction

The application is developed to manage Dipcons yearly invoices. This is done by extracting the relevant information from three databases, a postgresql 8.3 and two mssql 2000, using that information to calculate the yearly invoices, transforming the data to the right format and then storing the invoices back into the postgresql database for more testing and validation before migrating the data to the economic database, an mssql 2008.

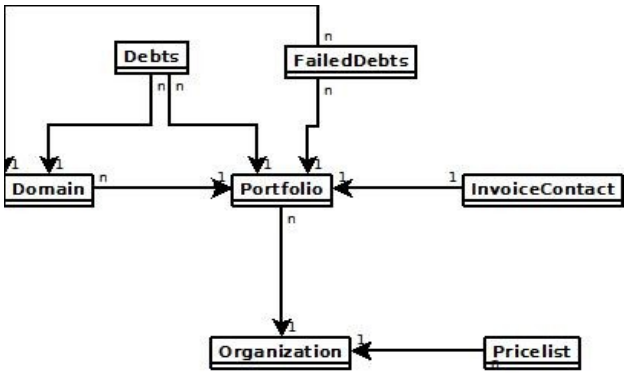
The application is separated into two independent parts, one that manages the invoice calculations with a web application as presentation layer and one that only handles the pure migration part of the service. That is due to that the second part can be used in combination with other applications and services so it benefits from being independent from the first part.

For the yearly invoices, there are three different types of fees that needs to be calculated; DMS fee, a domain name maintenance fee, REN fee, a fee for the renewal procedure for the domains and a HOST fee, a fee for the different hosting services used by the customer.

4.1.1 Database structure

There are four different databases involved in the application, all of them are relational databases. Three of them contains the relevant data needed for the application, and one of those three also stores the calculated debts. The forth database is the economy database to which the stored debts finally will be migrated and then become invoices.

The three databases that data will be extracted from is Dca2, a Postgresql 8.3, PNMS , a MSSQL 2000 and Domain which is a MSSQL 2008. The economy database, Portek, is also MSSQL 2008.



Picture 2. The key tables of the DCA2 database.

The databases maintains lots of data of which only a small part is relevant during this project, the tables of most interest in the DCA2 database are:

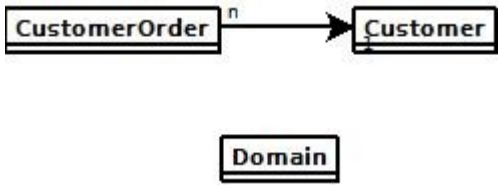
- Domain, from which the active domains are extracted.
- Portfolio, the connection between a domain and an organization.
- InvoiceContact, invoice information connected to the portfolio.
- Organization, the customer.
- Pricelist, an organizations individual pricelist.
- Debt, where the successfully created debts are stored.
- FailedDebts, where debts that couldn't be calculated are stored.

In order for a debt to be calculated, the relations between domain, portfolio, invoiceContact, organization and pricelist has to exist.

Picture 3. The key tables of the Domain database.

The tables of most interest in the Domain database are:

- TldInformation, which contains information about the type of the TLD, like if they are a ccTLD, gTLD and so on.
- TldInformationRenewalPeriod, which contains information about the renewal price for the different TLDs.

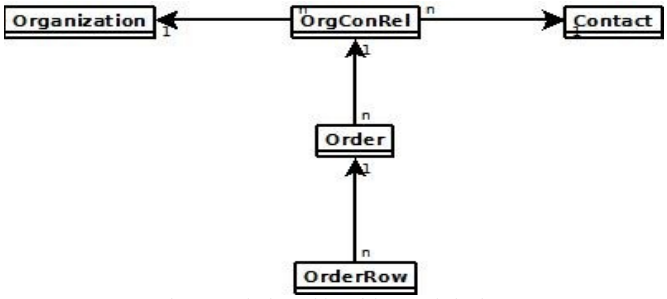


Picture 4. The key tables of the PNMS database.

The tables of most interest in the PNMS database are:

- CustomerOrder, from which Dipcons active hosting orders are extracted for calculating the HOST fee.

- Customer, the customer that owns a customerOrder.
- Domain, in which a domain Id from customerOrder can be translated to a domain name.



Picture 5. The key tables of the Portek databse.

The tables of most interest in the Portek database are:

- Organization, contains information about an organization.
- Contact, contains contact information about a person.
- OrgConRel, connects an organization with a contact and one or more orders.
- Order, contains information about an order/invoice.
- OrderRow, contains information for a row for an order/invoice.

The debts are migrated from DCA2 to Portek and the relations between Organization, OrgConRel, Contact and Order is required for an order to be valid.

4.1.2 Languages/Tools

The application has been developed in the Eclipse Galileo developing environment. Eclipse was chosen mostly because of its familiarity, and because of the knowledge that the required plug ins works well together with Eclipse.

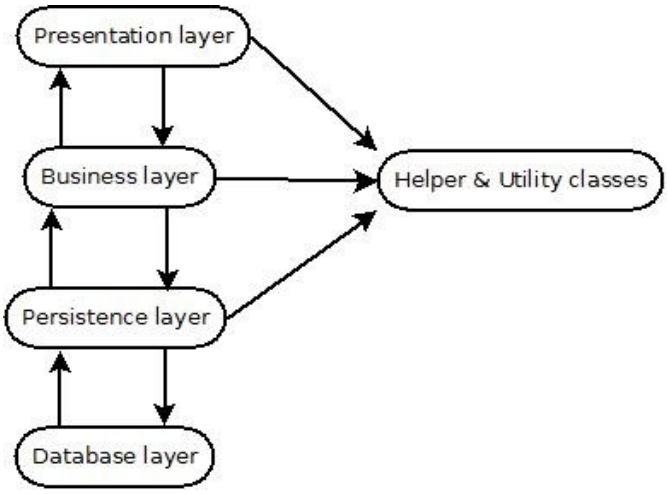
The plugin that has been used is TestNG as the testing framework. It were chosen after recommendations from the supervisor at House of Ports.

The language that the application is developed in is the Java programming language, this is after a requirement from the company. The persistence layer and databases is managed by Hibernate, an bject-relational mapping, ORM, library for Java and JBoss.

The queries to the databases is mostly coded in JPQL, but in some special cases they are in SQL. JPQL is used as a consequence of the use of Hibernate. When the persistence classes, the entities, are designed after the JPA, it follows naturally to use JPQL.

The web application, the GUI for the application, was developed using Vaadin, however, there was not time enough to complete it.

4.2 Design



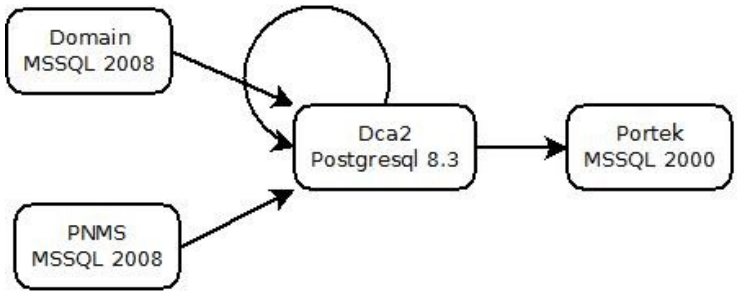
Picture 6. The layered architecture that has been used.

Both parts of the application is designed using layered architecture. That is an hierarchical architecture where each layer only communicates with the layer below and the layer above. In the application four layers have been used. There is a database layer, a persistence layer, a business layer and a presentation layer.

In the architecture there are also a layer, or a group, with helper and utility classes, like exception classes and data structure classes.

The persistence layers with Java entity beans has been developed ”bottom up”. That means that they have been developed from an existing database schema and data model as opposed to the database schema being developed from existing entities.

4.2.1 Database layer



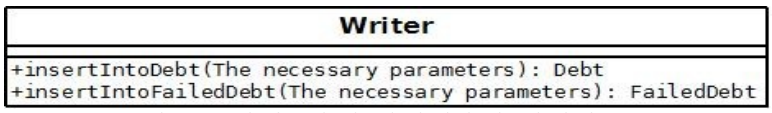
Picture 7. The four involved databases and their connections.

The database layer consists of the four involved databases, Dca2, Domain, PNMS and Portek. Data is extracted from Dca2, Domain and PNMS through the persistence layer,

treated by the business layer and then loaded back into Dca2, again through the persistence layer. When all debts are calculated and confirmed, they are migrated to Portek.

For the first part of the application, the one managing extraction and transformation of data, only the DCA2, Domain and PNMS databases is involved in the database layer. For the second part, migration and verification of data, only the DCA2 and Portek databases are involved.

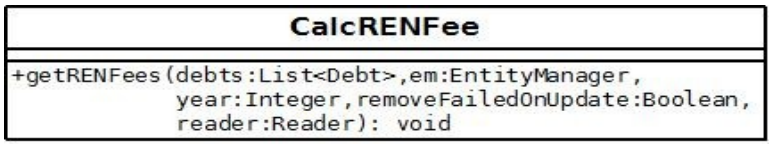
4.2.2 Persistence layer



Picture 8. Writer.java, the class that loads data into the database.

The persistence layer for each part of the application consists of the necessary entity classes representing the database tables involved in that part of the data migration process and two classes for extracting and loading data from and to the databases. An entity class is a class that represents a database table and contains parameters representing the fields in the related table and getters and setters for the parameters as well as syntax describing the relationships between the entities. A complete view of the persistence layers for each database is found in the class diagrams in the appendix.

4.2.3 Business layer



Picture 9. CalcRENFee.java, the class that calculates the renewal fees.

The business layer for the first application consists of four classes, one for each type of fee that needs to be calculated from the extracted data and one class that handles recalculations of fees that will be used after updates or fixes to possible inconsistencies.

In the second application the business layer consists of only two classes, the MigrationHandler.class that is the main class that maintains the migration of the data, and the CurrencyConverter.class that converts the currencies of the fees to the one required by the customer.

Class diagrams as well as sequence diagrams for both applications can be found in the appendix.

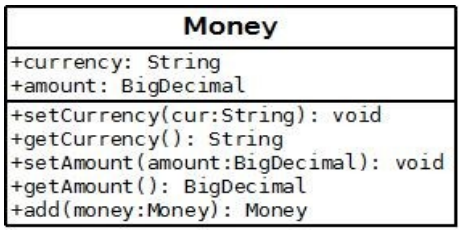
4.2.4 Presentation layer

Both applications are joined in the presentation layer, that is a web application developed using Vaadin. It was not completed on time, before the presentation of this thesis, so it is left out.

4.2.5 Helper and utility classes

Picture 10. An exception class.

In order to handle all the different exceptions that can arise while calculating the invoices some Exception classes has been developed. They handle exceptions like that if the domain name that should have a REN fee doesn't belong to any portfolio and/or organization in the database. The main way these exceptions are managed is that the domain in question is stored in a separate database table over failed debts together with an informative error message. Then the GUI will display a list of all the domains in this table together with their messages so that the inconsistencies can be corrected by the end user and the debts recalculated before the migration.



Picture 11. The Money class.

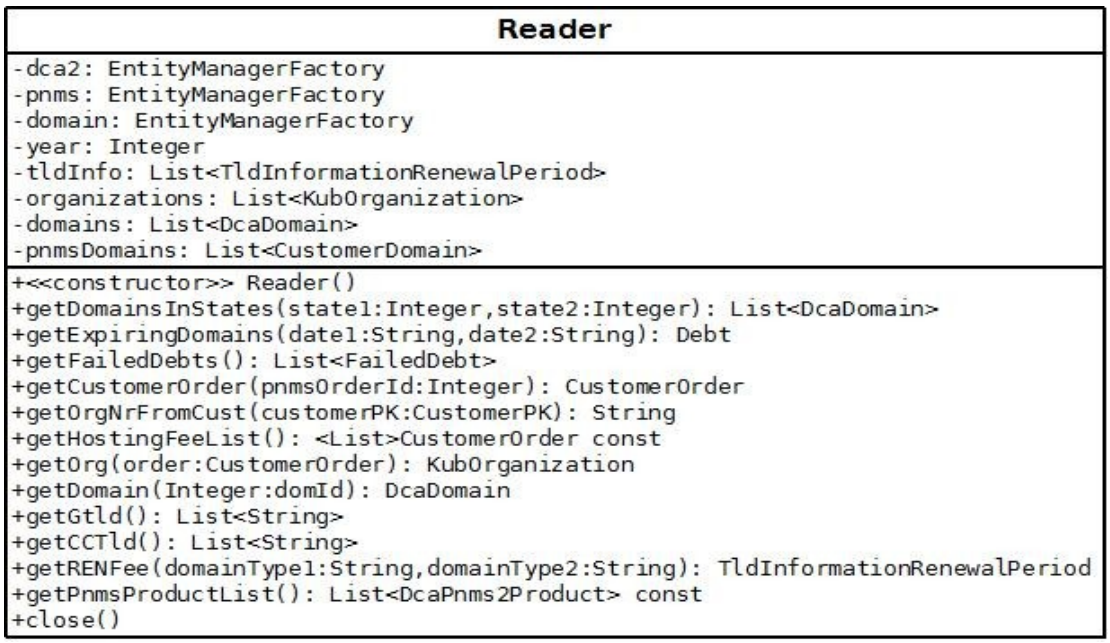
Another example of a utility class is the class *Money.java* that represents money with a field for the amount, a field for the currency and an add operation. This is useful when the fees shall be added together, so that no fees with different currencies get added before a currency conversion.

4.3 Functionality of the extraction and transformation of data (Step 1 & 2)

The functionality discussed in this part of the report is the functionality of the first part of the application, the invoice calculations. This functionality consists of four steps, extracting the data, calculate the fees, loading the developed data back into the database again and finally present the calculated debts through a web interface to be inspected and confirmed, and in some cases corrected. The migration part of the application is discussed under 3.4 Migration and more information about the user interfaces is found under 3.5 GUI. The class diagram as well as sequence diagram for this application is located in the appendix.

4.3.1 Extracting data

Extraction of the data is the first practical step to take during a data migration process, when the profiling of data and scoping has been completed..
The extraction of the data from the databases are done using Hibernate and JPQL. Various data is extracted from Dca2, PNMS and/or Domain, depending on which type of fee that is to be calculated. Depending on a call from the presentation layer, it is the classes in the business layer that initiates an extraction through the operations in the Reader.java class.



Picture 12. The class that manages the extraction of data from the databases.

Since there is a lot of common data that is treated several times for the various fee types, like domains and organizations, all of the calculation classes in the business layer uses the same instance of the Reader class which extracts the most relevant data when it is initialized in order to minimize the uptime on the databases.

In this case the constructor extracts:

- tldInfo, which is a price list for the diverse renewal prices for different TLDs.
- organization, a list of customer organizations, domains, a list of all the domains in the Dca2 database.
- domains, a list over all the active domains in the DCA2 database.
- PnmsDomains, a list over all the domains in the Portek database.

The EntityManagerFactories simply manages the connections to the databases. The Integer *year* is needed in order to know which years yearly invoices the application should handle so there won't arise collisions with previous years invoices that may be

left in the system.
The getter functions all extract data from the databases or reads some specific information of the already extracted data depending on which data that is needed at the moment.

The most important ones are the one selecting which domains that should receive a fee, and which hosting services that are active:

- getDomainsInStates, which returns the currently active domains (they are in one of two particular states) which should receive a DMS Fee.
- getExpiringDomains, which returns the generated debts that have a domain with an expiring date between date1 and date2 and thus need a renewal and a REN Fee.
- getHostingFeeList which returns all the active orders for hosting services performed by Dipcon.

In case a getter function that is supposed to retrieve some data fails to do so, a suitable exception is thrown.

The exceptions that the operations in Reader.java can throw are exceptions due to lack of data or lack of required relationships:

- CustomerToOrganizationException, in case the customer doesn't belong to any organization.
- NoDomainFoundException, in case a PNMS customer domain doesn't exist in Dca2.
- NoFeeFoundException, in case the TLD of a domain name doesn't have a renewal price.
- NoOrganizationFoundException, in case the noted Organization number on a customer order doesn't exist as a customer organization in Dca2.

In case of a thrown exception the domain or order in question is added to a list over debts that have failed to be calculated and then passed over.

In order to further reduce the uptime on the databases the fetch type on the ID fields in the entity classes is set to EAGER in order to get all the necessary information at the same time.

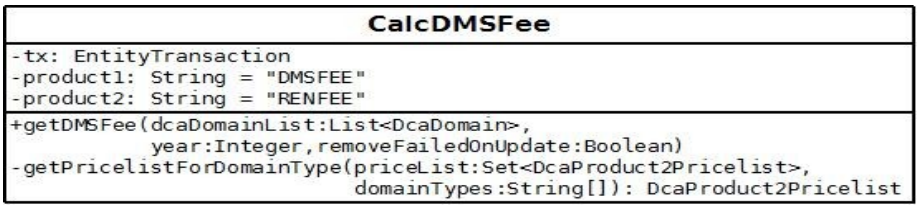
4.3.2 Fee calculations

There are three different types of fees that should be calculated and loaded for the yearly invoices. The three fees are DMS, REN and HOST fee. Each one is managed by a separate class in the business layer. In addition to them, the business layer also

contains a class that manages the recalculations of fees. The recalculation class is used to speed up the recalculations after a change in the databases that fixes one or more problems that caused a domain or order to become failed. Instead of rerunning the application and recalculate all of the debts, it only recalculates the debts that is stored as failed debts.

4.3.2.1 DMS fee

The first fee to be calculated by the application is the DMS fee. The DMS fee is a domain maintenance fee, so all the active domains should have a DMS fee connected to them in order to complete the yearly invoices. Each customer organization has a separate price list where the organizations different DMS prices is stored. The cost for the DMS fee can vary depending on the type of the TLD of the domain. One kind of distinction is if the TLD is a gTLD or a ccTLD, that is, if the TLD if a generic TLD or a country code TLD. In order to determine if a TLD is a ccTLD or a gTLD a list of all ccTLDs and another list of all gTLDs is extracted from the Domains database. These lists are returned by the operations getCCTld and getGTld in Reader.java.



Picture 13. The class CalcDMSFee.java.

The class that manages the DMS fee is named CalcDMSFee.java. It has two operations, the important one is getDMSFee, it calculates the DMS fee for each domain in the in parameter list of domains and stores it in the Dca2 database table Debts or Failed_Debts. An important thing to notice is that for a REN fee to exist for a domain, the domain must first have a DMS fee. Therefore, if the DMS fee calculation for a domain fails, so will the REN fee. Thus the REN fee for the domain will be added to the list of failed debts at the same time as the DMS fee for the domain is, if it should fail.

During the calculation process several consistency checks is performed in order to confirm that the databases contains the necessary information required for an invoice. In addition to that, the domain name is checked against a regular expression to confirm that the domain name in fact is an approved domain name. This check is performed by one of the helper classes.

The relations that needs to exist for the data migration to Portek:

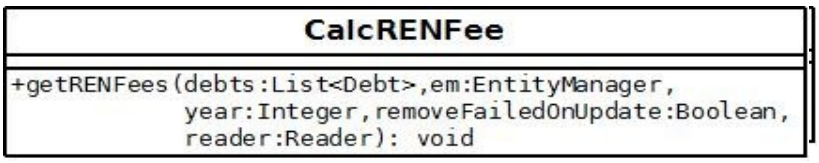
- The domain has to belong to one portfolio.
- The portfolio has to have an invoice contact.
- The invoice contact has to have a VAT-number.
- The portfolio has to belong to one organization

Also, for a DMS fee to be calculated, the organization has to have a price list which contains a relevant price for the TLD of the domain.

If a domain passes all the tests, a debt will be added, otherwise a failed debt will be added along with a comment about which test that failed.

4.3.2.2 REN fee

When the DMS fees are calculated, the application will continue with the calculation of the REN fee. As mentioned earlier, a REN fee can't exist without a DMS fee, so it is the domains that have a DMS fee that are interesting to work with. Therefore, the RENfee is generated from the collection of DMSfees stored as debts. In addition to that, the REN fee is the fee for the renewal costs for a domain, so it's only the domains that have an expiration date during a set time period that will be considered. It's the operation getExpiringDomains in Reader.java that extracts the debts that belongs to domains that are in question for a REN fee.



Picture 14. The class CalcRENFee.java.

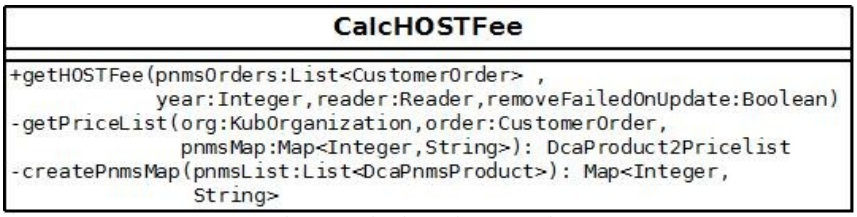
The RENfee cost is solely based on the type of TLD that the domain has. A price list with the particular price for a domain is returned by the operation getRENFee in Reader.java. The price list over the diverse TLD renewal prices is extracted from the TldInformationRenewalPeriod table in Domain database. If a renewal price doesn't exist in the database for the specific TLD, an exception will be thrown and the domain will be considered as a failed debt.

Since a REN fee can't exist without an existing DMS fee, the domains in question for a REN fee has already been checked for database and domain name inconsistencies, so the only reason for a calculation of a REN fee to fail is if the renewal price for the TLD of the domain doesn't exist.

4.3.2.3 HOST fee

The final calculation to be made is that of the HOST fees. Some examples of hosting services are DNS services, Email services, FTP service etc.

The HOSTfee calculation is the most complex one, since Dipcon offers several hosting services and each has it's own price. Contributing to the complexity of the task is also the fact that the HOST fee calculations requires a lot of data from both Dca2 and PNMS and that the data in the two databases in some cases is inconsistent.



Picture 15. The class CalcHOSTFee.java.

This requires several consistency checks, as well as a high amount of processed data, and due to the high amount of inconsistencies, the hosting fee dominates the failed debts. Due to this, the majority of the getters in Reader.java is developed for CalcHOSTFee.java.

Unlike the DMS and REN fee, the HOST fees are not based on domains, but on orders stored in the PNMS database, so called customer orders and each domain may have several hosting services connected to it. While the DMS and REN fees are calculated for each eligible active domain, the HOST fees are calculated one for each customer order. On these customer orders the name of the service, henceforth called product, is encoded as an integer. The decoding from integer to a descriptive name is done in a table in PNMS and accessible for the getHOSTFee operation through the createPnmsMap operation that maps each integer to a descriptive string.

Just as in the calculations of the DMS fees, the price list for the hosting services are linked to each organization, i.e. each organization has it's own price list of the hosting services they are utilizing.

Because of the high amount of data in the HOST fee calculations compared with the DMS and REN fee calculations, there are also more database inconsistencies that may arise:

- The integer product id on the order should have a related product name or product description.
- The domain id on the order needs to exist as a customer domain in the PNMS database, and a domain with the same domain name (i.e. the same domain) must also exist as a domain in Dca2.
- The Dca2 domain has to belong to one portfolio.
- The portfolio has to have an invoice contact.
- The invoice contact has to have a VAT-number.
- The portfolio has to belong to one organization
- The organization that the portfolio belongs to should be the same organization as the one listed on the order in PNMS.

In some cases the organization that the portfolio belongs to is not the same as the organization listed on the order in PNMS. That isn't necessarily a problem, but a note informing the end user of it should be added.

As in the other two cases, a fee needs to pass all these tests in order to be stored as a debt. If it doesn't, the order id will be added as a failed debt with a suitable comment.

4.3.2.4 Recalculation of data

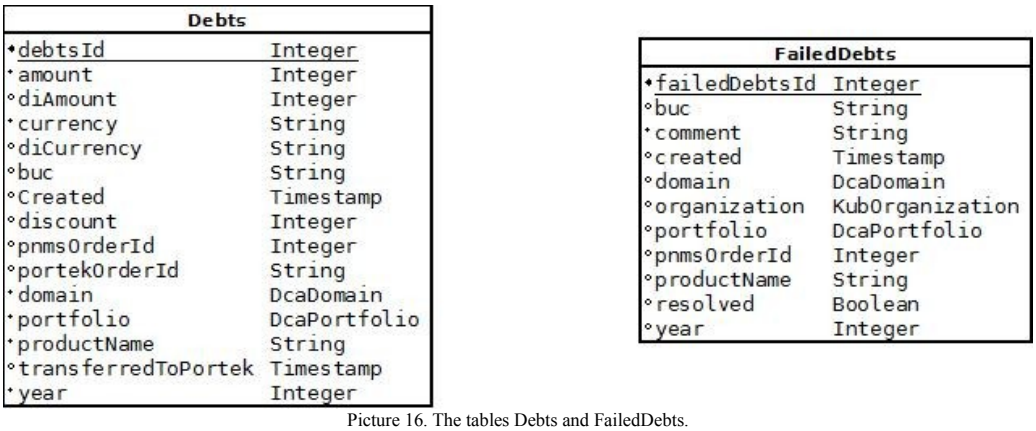
It is likely that some debts will fail to be generated and thus end up in the failed debts table. These failed debts are required to be corrected, that is; the data inconsistency has to be corrected, and recalculated before any migration takes place. The failed debts are presented in a web interface with a comment about what should be corrected for them to be calculated. When the correction has taken place, there is several possible strategies to recalculate the fixed failed debts:

- Recalculate everything by just rerunning the application. An easy way, nothing has to be added to the code, but it is very time consuming to rerun the application.
- Recalculate every failed debt. More work, but still nothing complicated. Less time consuming, since a lot less data is processed.
- Recalculate the failed debts that has been fixed. Most work, a way to recognize fixed debts has to be added, the GUI needs to contain more functionality. The required time for a rerun is down to a minimum.

The choice of strategy fell on number two, recalculate every failed debt. Due to the lack of time until the deadline for the project a compromise between running time and developing time was needed.

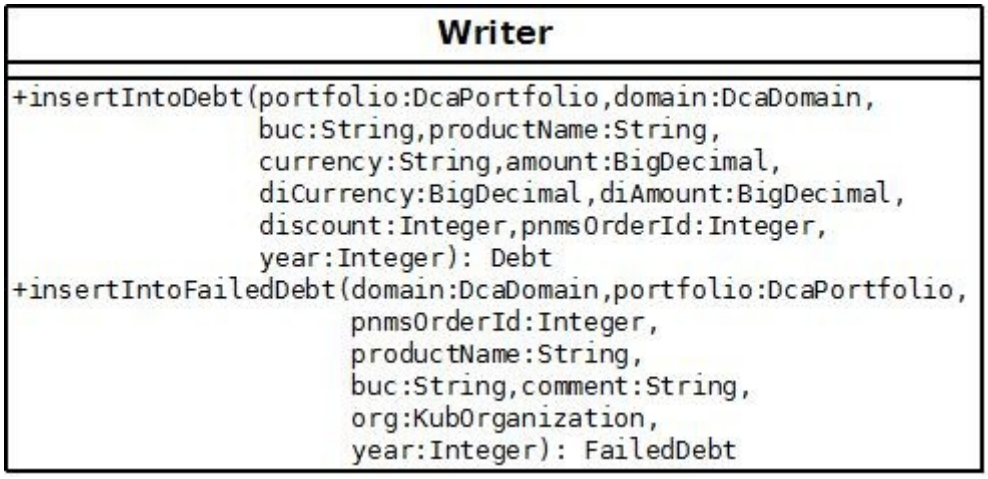
For the recalculations a new Reader is initialized, that extracts the required information for a recalculation from the databases so it's ready for the calculation. After that the original calculation classes are called with the parameter *recalculation* set to true. When the required information from the failed_debts table has been read, the entries are removed to make place if the calculations fail again on the same debt.

4.3.3 Loading data



Picture 16. The tables Debts and FailedDebts.

The data that is obtained during the calculation step is then loaded into the DCA2 database into one of the two tables Debts or Failed Debts. The debts that should exist but for some reason couldn't be calculated, most likely because of some database inconsistency, are stored together with the relevant information in the Failed Debts table. The Debts that are calculated and completed are stored in Debts. After an inspection they are ready to be migrated to the economy database and become invoices.



Picture 17. The Writer.java class, that loads data into the database.

It is the class Writer.java that manages the loads to the database Dca2. It has only two functions, one to load the data of the successful debts into Debts and the other to load the data of the failed debts into FailedDebts. The parameter data inserted into Debt are the necessary information for the synchronization with the economy database, Portek, as well as the raw debt information like currency and amount. The parameters to FailedDebt contains the needed information in order to identify which domain/PNMS order that caused the problem, as well as enough information in the comment to identify what the problem is. In order to avoid duplicates of debts when the application is run more then one time for a yearly invoice period, checks whether there already exists a debt with the current combination of year, product Name and domain (if the product name is DMSFEE or RENFEE) or pnmsOrderId (if the product name is one of the hosting services.) are conducted prior to the inserts.

To minimize the uptime to the database the debts are not loaded one by one, rather they are stored in memory until several debts have been calculated, and then the debts are loaded, persisted, to the database in a batch.

4.3.4 Presentation of the created debts

When the extraction, calculation and loading has been completed, the data from the tables Debts and Failed Debts are available for inspection through the presentation layer, the web interface. This function serves two purposes, one is to inspect and confirm that the calculated debts are correct and the other is to inspect the failed debts and see what should be done in order to make them pass the calculation step or to know that that invoice needs to be manually developed.

4.3.5 Testing

The framework used for the testing of the software is TestNG through the Eclipse Galileo development environment. The unit tests are focused on that the right data has been calculated and that the application creates all the applicable debts. (For each active domain/pnms order there should exist a debt). There are also tests that makes sure that the right exception causes the right comment in FailedDebts.

The most general test cases are run after the application has run, and makes sure that:

- For each existing domain in an active state, there is either a Debt with product name *DMSfee* or a FailedDebt with the same product name.
- For each existing domain in an active state that also has an expiring date during a certain time period their exists either a matching Debt or a matching Failed Debt with product name *RENfee*.
- For each applicable customerOrder in the PNMS database, there exists a matching Debt or FailedDebt, that has the same pnmsOrderId as the customerOrder.
- For each fee type there is also a check that Debts and FailedDebts combined doesn't contain more fees than the number of active domains, the number of active domains with an applicable expiration date or the number of applicable customerOrders, respectively.

For the more specific test cases, that tests whether the right data has been saved in the right Debts and whether the right exceptions causes the right comments in FailedDebts, the test program creates several test cases, some for Debt and some for FailedDebt.

The whole structure of domains that are contained in a portfolio, that has an invoice contact and belongs to an organization that has a price list is created in an controlled environment. Then the different test cases are tested, several tests where the data is consistent and the Debt should be created and several cases, at least one for each possible exception, where the data is inconsistent and a FailedDebt should be created.

The data tested for the Debts are:

- That the amount stored in the Debt is the same amount that the application should have extracted from the organizations price list.
- That the currency stored is the same as the one that the application should have extracted from the organizations price list.
- That the product name is the intended one. The product name is extracted from the price list for the DMS- and RENfees, and extracted from the customer order in the case of a HOST fee.
- That the domain Id stored is the correct domain Id.

- That the portfolio Id stored is the correct portfolio Id.
- For the REN fees, two types of amount is needed, and one currency for each, so in case of REN fee, both amounts and both currencies are tested.
- In case of a HOST fee, the PNMS order Id is tested as well, to make sure that it confirms with the order Id from the actual order.

For the FailedDebts the most important thing is to make sure that the correct comment has been added after an exception has been thrown. In addition to this, the test also confirms the data that is loaded into the FailedDebts. Since the data that is loaded into FailedDebts is varying depending on when the exception was thrown and what information that was available at that point the tests of the data is different for each exception. Information that may exist is information like domain name, PNMS order Id, organization and portfolio. Of course, if the data inconsistency is that the domain doesn't belong to any portfolio, the FailedDebt doesn't contain any information about either portfolio or organization.

In addition to the tests of the most apparent functionality of the application, there is also tests for the regular expression that confirms if a domain name really is a domain name. This is done by testing various different variants of domain names; names on “normal” form, IDN names, names in punycode etc.

Rules for domain names according to the DNS:

- No numbers are allowed in the TLD.
- The TLD is between 2 and 6 chars (besides for punycoded domain names).
- A domain name has to begin with a letter or a digit.
- A domain name can not contain any other char then digit, letter, '.', '-'.
- A domain name needs to contain at least one dot.
- The TLD has to be an existing TLD, the list of all TLDs is extracted from the database Domain.
- A domain name has to be less than 253 characters.
- Each label in the domain name has to be less than 63 characters.

4.4 Functionality of the transformation, migration and verification (Step 3 & 4)

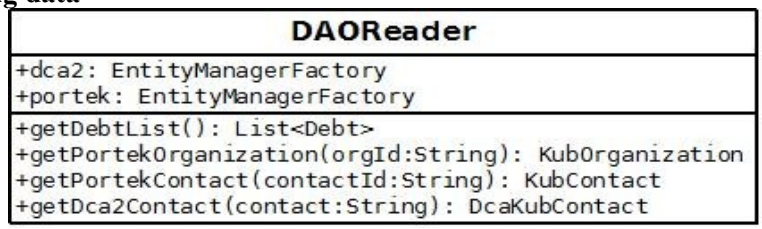
This part of the application manages step three and four in the data migration, loading and verification of data. In order to achieve the right currency on the invoices, and the same currency for all fees that are on the same invoice, some currency conversion takes place as well.

The application manages the migration of the calculated invoices from Dca2 to Portek

as well as the necessary transformations. The data migration is a migration between two databases, from a postgresql 8.3, to a MSSQL 2000.

A class model and a sequence diagram of this application is available in the appendix.

4.4.1 Extracting data

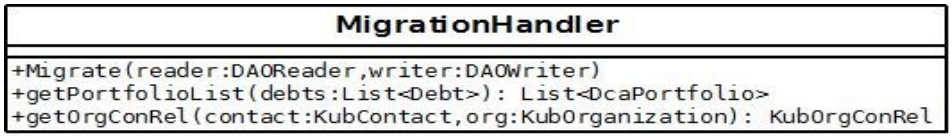


Picture 18. The DAOReader.java class.

The DAOReader.class extracts data from the Dca2 database and the table Debts, which contains the completed debts. In addition to the debts, the information about organization, portfolio, invoice contact etc. is extracted, since it is needed in Portek, the economy database, as required information in the invoices.

DAOReader also extracts some data from the Portek database, organizations and contacts. That is due to that the relation between organization, contact and order is required for an order to be legit. If the customer organization or contact doesn't exist in the Portek database, it has to be created. The only purpose is to make sure that the relation exists, if else, create it, and to get the id for the relation, in order to connect the order to it.

4.4.2 Transformations



Picture 19. The MigrationHandler.class.

When the debts has been extracted from the database along with the rest of the required information, the transformation from the Debt format to the Order format takes place. This is, together with the debt calculations in the previously discussed application, part two of the data migration process, the transformation of data.

The orders are sorted by the portfolios from DCA2, all of the debts that belong to the same portfolio are transferred to the same order. Therefore the operation getPortfolioList returns all the portfolios involved in the debts. They are then iterated through, generating one order for each portfolio, with one order row for each debt.

When executing this step the required information exists in the DCA2 database, but not necessarily in Portek. If the customer organization, or the contact, doesn't exist in the Portek database, they have to be created. However, the necessary information for creating them exists, as pointed out earlier, in the DCA2 database, so it's a simple

process of extracting and loading data. A relation between the organization and the contact is required as well, but is as simple to create if it doesn't exist.

The layout of the orders is designed by the economy system, Visma, that sorts the row of the order on basis of product id and name. Therefore it is important to follow the standards used in Visma, in order for the rows to appear correct on the invoice.

4.4.3 Currency conversion

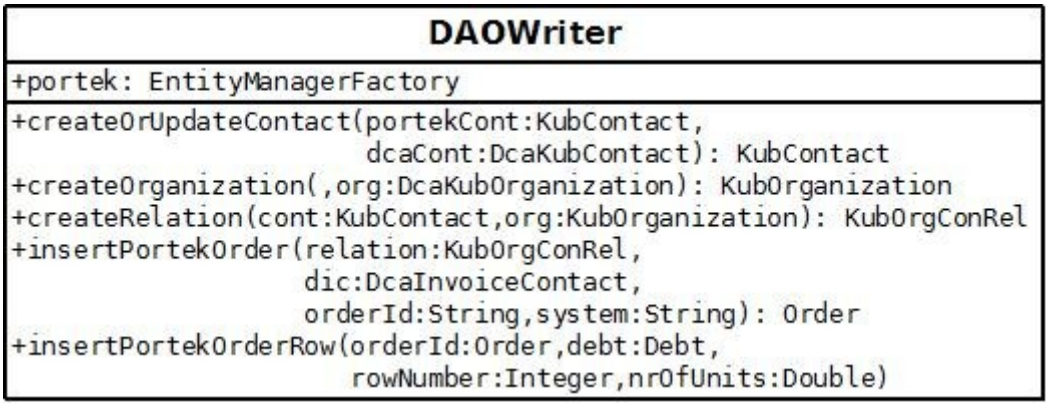
Each debt may have it's amount in it's own currency, and before several debts can be added to the same invoice, the currencies has to be the same. To prevent debts with different currencies to be added to the same invoice, the Money.java data structure is used.

The currency conversion is extracting up to date data about the currencies from a web service before the conversion takes place. The currency for the invoice is dependent on the choice of the customer organization.

The currency conversion is primarily done through the use of an existing web service and .wsdl file and javas wsgen.exe file. On top of those classes is the main currency converter class that maintains the logic, requests the currency rate and then multiplies the amount of the invoice with that rate.

A class diagram including only the currency conversion part of the application is included in the appendix.

4.4.4 Loading data



Picture 20. The class that manages the loading of data to Portek in form of orders and order rows.

When the transformation of the debts into orders and order rows are completed with the right currency, they are loaded into the Portek database as batches.

The DAOWriter.class also manages the creation and loading of nonexisting organizations, contacts and “organization to contact” relations. They are simple operations that merely copies the information from the supplied organization/contact from the Dca2 database.

4.4.5 Verification

After a completed migration of the data to the new system, an independent validation of the data is performed, in order to ensure that the data integrity is intact and that there is no data losses. In order to ensure that the validation is independent from the migration architecture, the validation is developed by another programmer.

4.4.6 Testing

The testing of the data migration part is similar to the testing of the data transformation part. This is done by going through all the portfolios and their debts, ensuring that each portfolio has resulted in an order, and that no other orders has been created, and ensuring that each debt has resulted in an order row, and that no other order rows has been created that contains a cost (i.e. non cosmetic rows). The values in the rows are being verified as well, by comparing them to the values in the respective debt table.

The values transferred to the order and the order rows are tested as well, the values tested for the orders are:

- That the currency is the expected one.
- That the billing information is correct.
- That the customer and seller references are correct.

The values tested for the order rows are:

- The order reference, that contains the id of the debt that has generated the row.
- That the three first rows contain information about which portfolio the order belongs to.
- The price is tested to make sure that it has been converted to the right currency.
- The product name is tested to make sure that it's the right one, and that it is a VISMA product name.

4.5 GUI

The graphical user interface intended for the application is a web 2.0 interface developed using Vaadin. However, there was not time enough to complete the GUI before the presentation of this thesis.

4.6 Encountered problems

There have been several problems encountered, and a majority of them solved as well, during this project. The majority of the problems are related to the databases and the languages used, but some of them has other sources. The problems that couldn't be solved in any reasonable time was simply worked around.

Several things that were done during the data migration project were not exactly problems, but could have been done much more smoothly with more knowledge about the data migration theory and practice. The key part were the data profiling, that took up almost a majority of the time spent on the project. With a more thoroughly data profiling phase at the start of the project, possibly using landscape analysis, the time spent on data profiling could have been properly reduced. The lesson learned is that it is better to do an exhaustive data profiling at the start, then several deficient data profiling phases during the course of the project.

The same goes for the planning of the project, it is too easy sometimes to just do a plan consisting of best guesses and some experience, instead of on a consistent investigation on the subject at hand. More time spent on planning is not equal to more time spent on the project, but the contrary.

4.6.1 Database inconsistencies

The problems that has led to the most frustration is the database inconsistencies. This was especially hard in the beginning of the project, since it was difficult to know if an encountered problem really was a database inconsistency or if it was something else. With more knowledge about the databases, their structures and which data that was required for the different parts of the application, the database inconsistencies became less problematic.

The database inconsistencies are not limited to the lack of required information, but also to the database structures. Tables with missing primary keys and without necessary foreign keys caused some trouble, especially with the use of JPA, that requires primary keys in the table entities and if there is none in the database one will be automatically chosen when generating Java entities from the database. Before this was realized it give rise to some pretty interesting errors in the code, when the wrong primary key was selected and because of that a lot of data “disappeared” (All fields with the same value in the chosen primary key were reduced to just one field).

When the problem was found the new problem that arose was to find a suitable field in the tables that could be thought of and used as a primary key without any disappearing data.

For the missing foreign keys, it was easy to create a foreign key mapping in the entities using JPA. The major problem was to harmonize the data, since the data loaded into these fields that should have had but not had a foreign key, was not matched as foreign keys with the rest of the database, some inconsistencies in between them was found and had to be corrected.

4.6.2 Language related problems

A very general problem that emerged several times during the development of the application was an issue with the JPA implementation. A JPA implementation with Eclipselink lacks the ability of informative error handling. Therefore various errors was encountered without getting the information needed in order to correct those errors. Due

to the lack of information, those errors where very hard to solve and the way to deal with them was mostly to work around them. I still have no idea what caused several of the errors reported, while I got some clarity with others when I noticed what roundabout that solved them.

4.6.3 Other problems

A problem that can't be thought of as either a language related problem or a database problem, is the size of the batches. When the DMS- REN- and HOST-fees are being calculated, they are not loaded into the database one after another, but several are calculated before they are loaded into the database as a batch. The problem was to find the most suitable size of the batches to minimize the uptime with the database as well as not having to store too much information about the debts in the memory, if anything goes wrong with the loading or any other part for that matter, the whole batch is discarded and has to be recalculated. The method used to develop the right batch size is practical tests with different sizes to see how they effect the running time of the application. As long as the new batch size gave a better running time, the new batch size was considered the best so for, when the running times got longer the batch size was reset to the last one that gave the best time.

Another problem, that hasn't to do so much with the coding but still is important to notice, arose because of the frequent contact with the non technical staff at Dipcon. While I got used to see the databases as pure data in fields in tables, the staff that actually works with the data and that are going to use the application observes the data through different graphical user interfaces and sometimes it was difficult to communicate and understand this difference.

An example is the data needed for the HOST fees, in a discussion about which data I should base the calculation on, I was shown the data in the graphical interfaces, so I still didn't have any clue about where in the database the data was stored. Thankfully, my supervisor at House of Ports had the knowledge about how the data shown in the interfaces correlated to the data in the databases and so he could help me.

This is important to notice because you work seldom alone while developing an application, more often a non technical customer is involved with requirements and opinions and it is important to understand each other, or if not, understand that you dont understand each other.

6. Discussion & conclusion

Data migration is a process that most companies bumps in to sooner or later. Although it might be tempting to postpone the first data migration due to all the risks involved, once it is done most companies perform data migration on a regular basis. With 60% of the interviewed companies in Bloors research[2] performing a data migration of some kind once every 3 month or more often, and a clear majority of all data migration projects fail to be completed in time and/or on budget, it is easy to see a need of more understanding of the data migration process. There is several different steps to take in order to increase the likelihood of a successful data migration. The first step is to recognize the data migration project as a project in itself, that it isn't just a step during the upgrade of software. Data migration is a complex task that is often misjudged. After recognizing the complexity of the task at hand, the next step is to profile the data and to scope the project. This step is usually overlooked, the one major reason while data migration projects fail is the lack of scoping. However, once the developing team and the client have recognized the complexity of the project, there is a broader understanding of the need of an accurate scoping and data profiling. The third step that is vital to remember is the contact with the business people, the people that will be the end users of the new system and are the ones that have knowledge about how the system works. To keep it simple, using a data migration methodology, like the one described in the previous section, and strive to follow it, is the most effective way to increase the likelihood of a successful data migration.

A data migration application can be developed in house and from scratch, as in the practical example, but since data migration projects are common there exists off the shelf products with different focuses that can be useful. The time and economical costs involved during a data migration, regardless if it is custom made or off the shelf, is often small in comparison to the costs that arise from a failed data migration project. During an evaluation of the available off the shelf products and/or data migration companies and there costs, it is of essence to investigate the consequences of an overrun of time and budget during the project. There are hidden costs like bad reputation for the company if the service is down, or if the company still uses an old version of the software a long time after a new one is available.

That which took most time during the development of the data migration application was to get the knowledge about the databases and systems. The time spent could have been decreased if there had been a landscape analysis at the start of the project, instead of a continues increase in the knowledge after hand, when it was required. A landscape analysis would have given a broader overview of the databases and systems, and would have helped avoid some of the troubles that arose during the project. Since there wasn't any knowledge about the scoping problem at the start of the project, a more exhaustive data profiling period was neglected.

The major advantage of developing a custom made solution in house is the close contact with the end users of the system. As soon as any questions or reflections arose they could be discussed and answered before continuing the development. One thing to keep in mind is that the business people may have little, or no, experience of the technical

work and that it is easy for misunderstandings to arise from the different views of the system. It is always better to spend some time making sure that everything is clear, then to take a chance and risk spending time developing something unnecessary or unwanted.

No off the shelf products were used during the data migration project, and since the mission from House of ports was a custom made solution, there wasn't any investigation around possible existing solutions. Some of the choices that has to be made, like programming language and if to split the migration, had already been made, or at least there were some preferences about them. This meant that there was no real evaluation about which language that was the most suitable or if the solution would have been simpler or had taken less time without a split. A split was needed for other reasons than ensuring a smooth and secure migration.

Due to a lack of time at the end of the project, the graphical user interface for the application was not finished, and neither was the independent validation. However, with the new knowledge one can see that the benefits of another programmer completing the independent validation part.

References

[4] Behm, Andreas, Geppert, Andreas & Dittrich, Klaus R. 1997. *On the migration of relational schemas and data to object-oriented database systems.* Switzerland. 5th international cnference on Re-Technologies in Information Systems.

[7] Callan, Steve. *Database migration – A planned approach.* Available at Http://www.databasejournal.com/features/oracle/article.php/3586516/Database-Migration-A-Planned-Approach.htm quoted 23/5 2010

[8] Callan, Steve. *Database migration – It's more than running exp and imp.* Available at Http://www.databasejournal.com/features/oracle/article.php/3586516/Database-Migration-It's-More-Than-Running-Exp-And-Imp.htm quoted 23/5 2010

[6] Carreira, Paulo & Galhardas, Helena. 2004. *Efficient development of data migration transformations.* France. SIGMOD.

[10] Davis, Norma,. *Six misconceptions about Data Migration.* Available at http://www.datamigrationpro.com/data-migration-articles/2008/2/4/six-misconceptions-about-data-migration.html quoted 24/5 2010

[1] Garcia-Molina, Hector, Ullman, Jeffrey & Widom, Jennifer. 2009. *Database Systems, the complete book*. 2nd Edition. Stanford. Upper Saddle River:PrenticeHall

[5] Hall, Joseph, Hartline, Jason, Karlin, Anna R, Saia, Jared & Wilkes, John. 1998. *On Algorithms for Efficient Data Migration.* USA. Hewlett-Packard Research Laboratories.

[2] Howard, Philip & Potter, Carl. 2007. *Data migration in the global.* United Kingdom. Bloor Research.

[3] IBM Global Technology Service. 2007. *Best practices for data migration.* USA. IBM Global Technology Service.

[12] Informatica. *Achieving a successful data migration.* 2010. USA. Informatica.

[9] IT Cortex. *Statistics over IT projects failure rate.* Available at http://www.it-cortex.com/Stat_Failure_Rate.htm quoted 1/6 2010

[15] Jones, Dylan. *Data migration project checklist – a template for more effective data migration planning.* Available at http://www.datamigrationpro.com/data-migration-articles/2008/12/3/data-migration-project-checklist-a-template-for-more-effecti.html quoted 23/5 2010

[13] Pick, Gershon. *Data migration Concepts and Challenges.* March 2001

[14] Platten, John. *The problem with data migration scoping.* Available at http://www.datamigrationpro.com/data-migration-articles/2008/3/5/the-problem-with-data-migration-scoping.html quoted 24/5 2010.

[11] *What is landscape analysis and why is it important to your data migration?* Available at http://www.datamigrationpro.com/data-migration-articles/2008/11/24/what-is-landscape-analysis-and-why-is-it-important-to-your-d.html quoted 24/5 2010