Data Migration Strategy Guide

Author:	Jerry Bryan Infomig Ltd
Document Version:	r1
Document Status:	Release
Document Date:	19/06/2009

Contents

1 Revision History 3
2 Purpose and Scope 3
3 Executive Summary 4
4 Introduction 5
5 Scoping 7
6 Planning 9
6.1. Overview 9
6.2. Requirements Gathering 9
6.3. Specification Creation 10
6.4. Data Extract and Cleansing 10
6.5. Data Transform and Load 10
6.6. Acceptance Testing 11
6.7. Go Live 11
7 Specification 12
7.1. Requirements Specification 13
7.2. Design Specification 15
8 Implementation 17
8.1. Development Environment 17
8.2. Data Extraction and Cleansing 18
8.2.1. Domain Checking 19
8.2.2. Enforcing Integrity Constraints 20
8.2.3. De-Duplication 20
8.3. Data Transformation 21
8.3.1. Mapping 22
8.3.2. Filtering 22
8.3.3. Separating 22
8.3.4. Combining 22
8.4. Data Loading 23
8.5. Testing 24
8.5.1. Test Plans 24
8.5.2. Data Extract Testing 24
8.5.3. Data Transformation Testing 25
8.5.4. Data Load Testing 26
8.5.5. User Acceptance Testing 27
8.6. Go Live 29

1 Revision History

Version	Author	Date	Changes Made
R1	Jerry Bryan	19/06/2009	First Release

2 Purpose and Scope

This document is intended to provide guidance to organisations who are contemplating migrating information from existing database applications into a new database application. It contains high-level planning and strategic guidance rather than technical details, and as such is aimed at the implementation project manager and the data migration manager.

Accompanying Microsoft Word and Microsoft Project documents are available from **Infomig** which provide templates for the data migration implementation. These are listed in the references at the end of this document.

Experience has shown that achieving a successful migration within the overall time constraints of an implementation is difficult unless it is properly planned and managed from the outset. This document seeks to make project manager aware of the difficulties and provide recommendations which will enable the migration to be completed on time, within budget and to the satisfaction of the users of the information being migrated.

3 Executive Summary

- Decide at the outset of the project which applications currently hold the data to be migrated and what data will be migrated from them.
- Treat data migration as a subproject of the overall implementation and proceed through the same phases and milestones.
- Monitor the progress of each phase to ensure that enough time is left for subsequent phases.
- Make generous allowances in the project plan for data cleansing and testing.
- Recognise the reporting issues that can be affected by data migration choices.
- Produce a requirements specification that is signed off by the user representatives before starting the design specification.
- Do not migrate data directly between databases. Produce logical data models of the existing applications and the new application and use the requirements specification to transform the data between those models. Build the business rules of the application into the logical application model.
- Perform the data extract and load between the physical and logical data models to check for data quality.
- Test each process separately and ensure that correct numbers of records are being migrated at each stage.
- Get users to perform acceptance testing of the migrated data, and allow for three rounds of acceptance testing.
- Plan for the switchover from the old applications to the new and decide how to record new data that arrives during the switchover.

4 Introduction

Database applications have been in common commercial use since the early 1980s[1]. They are used whenever a business needs to store and retrieve information about its clients, its products, its suppliers and even its staff. Almost every medium and large size organisation now has at least one database application and these applications often hold information for the whole business.

These database applications have come to be relied upon to such an extent that a computer system failure will halt the functioning of the business. Companies now recognise how important database applications are, and put in place elaborate and expensive disaster recovery systems and strategies to ensure that the information that is vital to their business is always available.

Because of the rapidly improving technology for computers and changes in business requirements the normal lifecycle for a database application is between 5 and 10 years. After this time the application's ability to store, process and retrieve data will not match the needs of the organisation or the technology on which it operates will have become obsolete. The organisation will therefore be required to implement a new database application.

What is usually not realised by companies when they replace a database application is the distinction between **information** and **data**. Although the users of an application 'see' their information in the way it is presented by the application, that information actually exists independently of the application and can be held in multiple applications or 'moved' from one application to another. However the way in which the same information is held in different applications is often different; data is the way in which information is held in a *particular* database application

For example a client will have a name and an address. In one application the client's first and middle names are entered into one field on a screen and in another they are entered into two separate fields. The information that each application stores for the client's name is the same but the data that each stores is different. Similarly in one application a client's address may be entered as 4 or 5 lines without any structure, whereas in another the building name, street number, street name, town, county and postcode may all have to be entered into separate fields. Again the address *information* held by each application is the same but the address *data* is different. These examples are fairly trivial but the relationship between information and data can be extremely complex.

This movement of information between database applications is usually referred to as **data migration**, but I prefer to use the term information migration to emphasise that there is much more involved than simply copying data from one database to another. The choices are therefore to input the existing information manually into the new application or to use electronic methods to extract the data from the existing application and transform and load it into the new application.

[1] I am referring here to applications built specifically to enter and retrieve information from a database management system (DBMS), as opposed to applications which store and retrieve data solely for use by that application.

For a medium or large sized organisation the amount of information they hold will prohibit manually entering it into a new application on the grounds of the resource time and cost required. Therefore an electronic method will be required which involves *extracting* the information from the existing application, *transforming* it into the format required by the new application, and finally *loading* it into the new application.

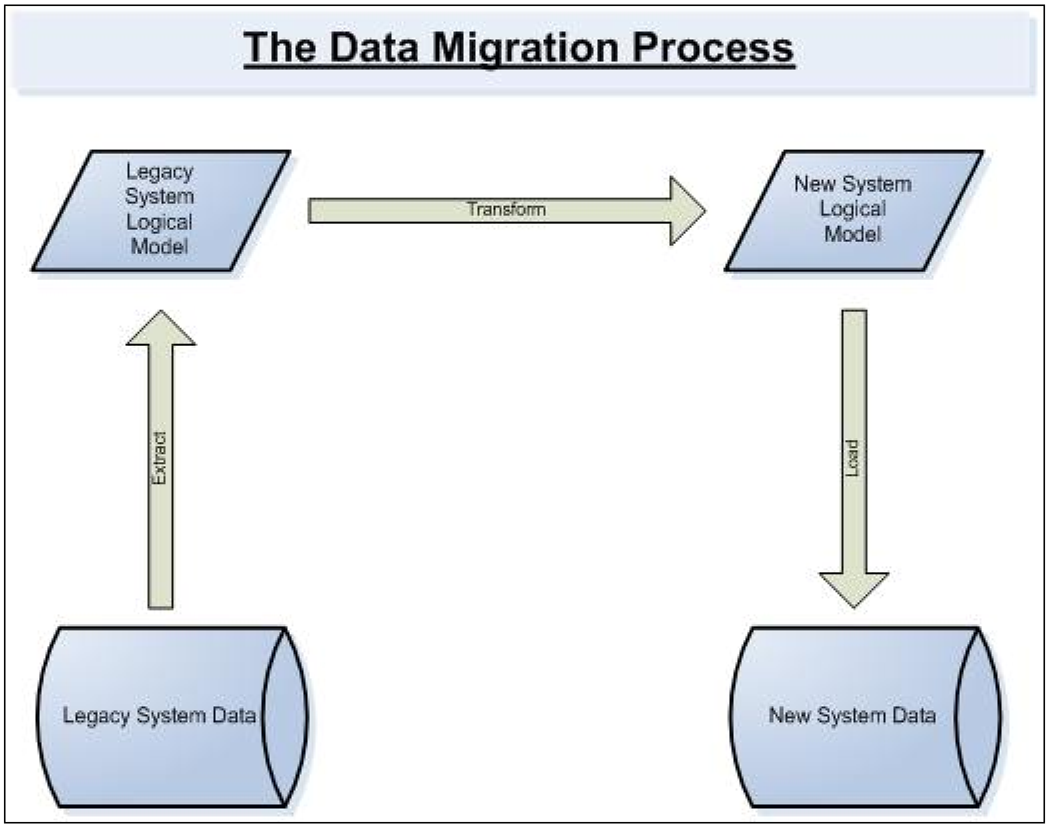

Figure 1: The Data Migration Process

These migration tasks are not usually trivial and so need to be scoped and included in the implementation project plan from the outset. If there are constraints on time and resource, as there usually will be, early decisions will need to be made regarding which existing information should be migrated and which should be left in its current format and location.

The remaining sections of this document will cover:

- Deciding which information should be migrated (scoping)
- Determining the resources and time required (planning)
- Mapping the information from the existing applications into the new application (specification)
- Cleansing and extracting the data from the existing sources and transforming and loading it into the new database (implementation).

5 Scoping

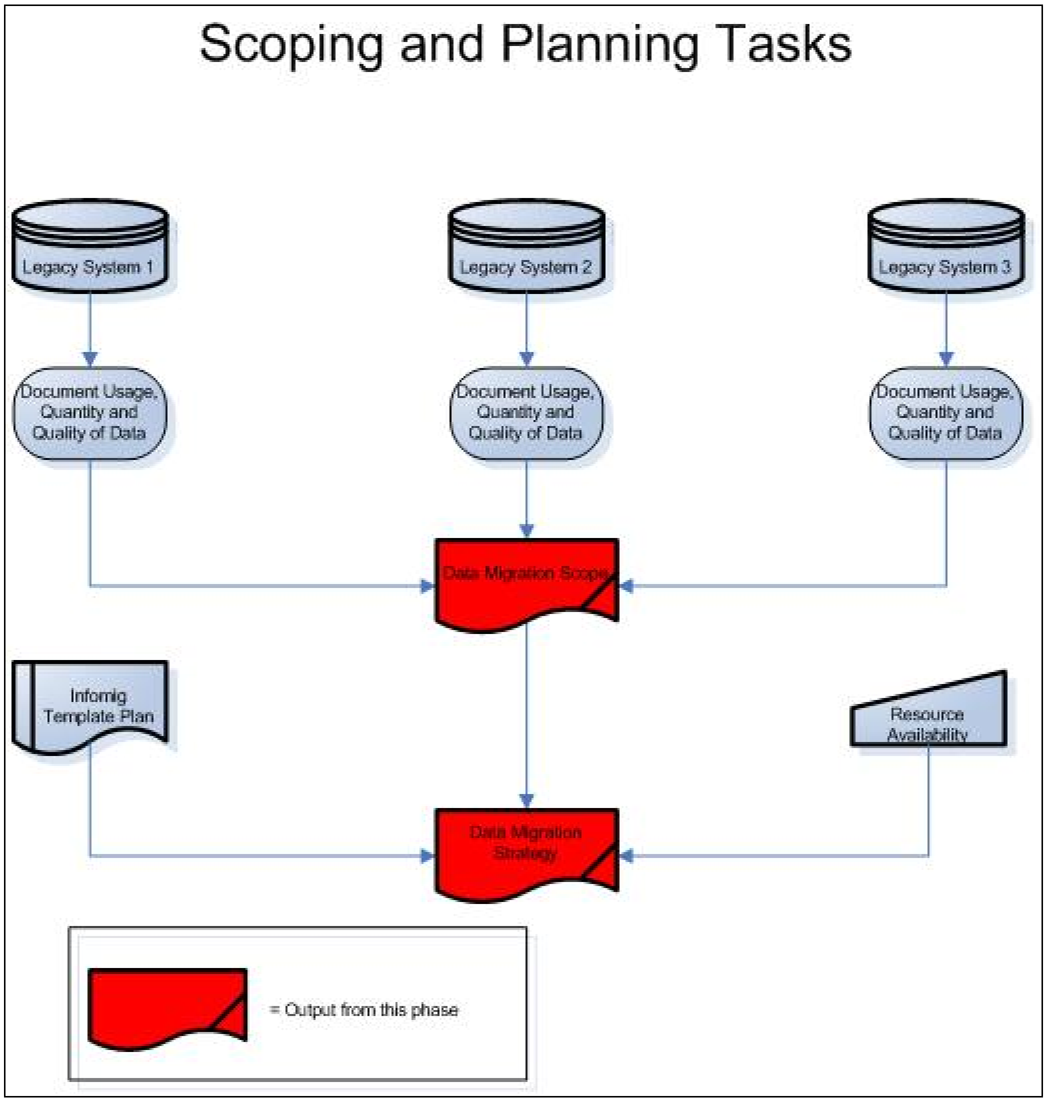
Scoping and Planning Tasks
Legacy System 1
Legacy System 2
Legacy System 3
Document Usage, Quantity and Quality of Data
Document Usage, Quantity and Quality of Data
Document Usage, Quantity and Quality of Data
Data Migration Scope
Infomig Template Plan
Resource Availability
Data Migration Strategy
= Output from this phase

Figure 2: Scoping and Planning Tasks

The first exercise that should be undertaken is to determine the sources of existing data that could potentially be migrated into your new application. This should take no more than 10% of the total time available for the implementation.

It is assumed that the person (generally the implementation project manager) who undertakes this task has a reasonable understanding of the information that can be held in the new application. The project manager should familiarise themselves with the current functionality of this application and should obtain the specification for any further functionality that is due to be included before the application is implemented.

If the implementation is being done in a number of phases then it is usual for the data to be migrated from existing applications in the same phases as the implementation. For example if a finance module is being implemented in a later phase it is not generally sensible to try to migrate data for it during the first phase. However consideration must be given at the outset to how data migrated in a subsequent phase will be integrated with data migrated in an earlier phase.

Once the project manager knows which information will be used in each phase of the implementation he or she should find out how that information is currently being held. At this stage it is not necessary to go into detail of how the data is held; the object of the exercise is to find out where it is being held and the feasibility of migrating it into the new application.

For each application identified a decision should be taken as to which information should be migrated from it and in which implementation phase it should be migrated. Factors that should be taken into consideration include:

- What is the information currently being used for, and is this function to be carried out in the new application?
- Will the existing application still be available and maintainable after the new application has gone live?
- How much of the information is historic, and how much of that historic information is still required to be held?
- What is the quality of the data and will it be possible to cleanse it?
- Could the data be manually entered into the new application and how long would this take?
- Is the same information duplicated in another application, and if so which application will be the primary source for data migration?
- Does the information in the application get transferred into or out from any other applications?
- Does the information get used for producing statutory reports or management information?

A scoping document should be produced which collects these decisions together into a Data Migration Strategy.

This document should be reviewed and signed off by all interested parties, especially representatives of the users of the existing applications. The decisions made in this document are vital in determining the project plan for migration and ensuring that time and resources are focussed in the correct areas. It will also ensure that there are no last minute requests by users for ‘their’ data to be included in the migration.

6 Planning

6.1. Overview

Once the decision has been taken as to which information should be migrated the new application and when the next step should be to identify the resources that will be required and to estimate the time that the process will take. This information should be pulled together in a data migration implementation plan. **Infomig** can supply a sample data migration implementation plan[i] and project template[ii]

The data migration process can be considered as a sub-project of the full implementation project, and it should proceed through the normal project phases and be subject to the same milestones as the rest of the project. At each milestone the risks and assumptions can be reviewed. The outcomes of these reviews should feed back into the project plan and may result in a change to the scope of the data migration. For example it may become apparent that the quality of the data from a particular source is too poor to justify the effort required to clean it.

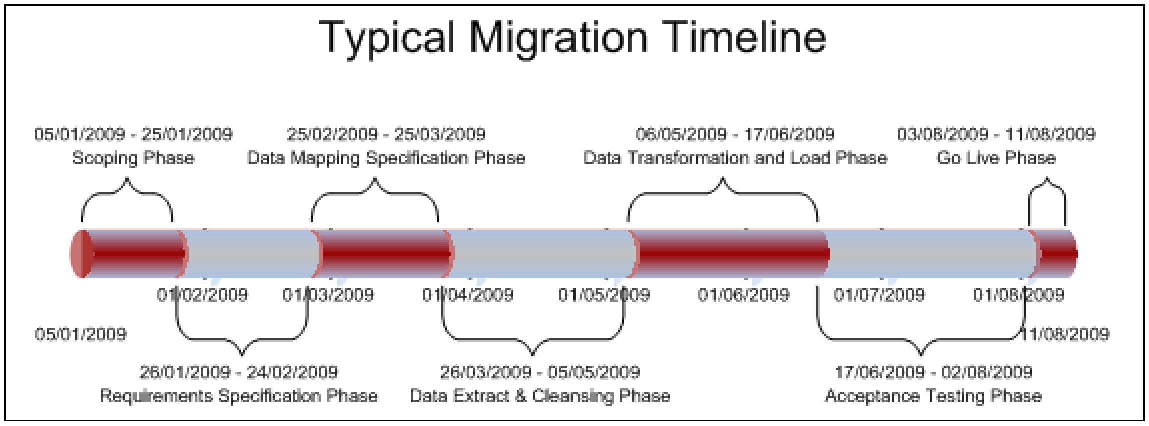

Figure 3: Typical Data Migration Timeline

6.2. Requirements Gathering

The first phase of the process will be to specify in detail which information will be taken from each of the existing applications and where it should appear in the new application. This should be a fairly high-level document written in business terms and aimed at the department managers. Its author would normally be a business analyst who has an understanding of the source applications and the new application. This information could be collected as part of the overall business process mapping exercise that will be carried out to determine how the new application will be implemented. The time required to produce this document will depend on the number of separate data sources and the amount of information in each, and would normally be expected to take around 15% of the total time available.

6.3. Specification Creation

The next phase will to be to draw up a technical specification of how the data in the current applications will be migrated into the new application. This document will be used by the people who will carry out the data migration. It will be produced by a database analyst / system designer who understands how data is held in the source databases. At this point a decision should also be taken regarding the development environment for the migration. The considerations for this decision are covered in the implementation section of this document. This phase will normally take around 15% of the total time available.

6.4. Data Extract and Cleansing

The creation of the technical specification should indicate the likely areas for data cleansing. Data cleansing involves checking the quality of the data in the source application and if necessary altering it so that it conforms to the intrinsic data integrity requirements of the new application. It may also be necessary to remove duplicate data and ensure data matches across the separate sources that are being used for migration. This work will normally involve users who have a good understanding of the data in the source applications and the ‘real-world’ knowledge of how that data is obtained.

The data cleansing work will normally be done in parallel with the development of the data extraction scripts. These will usually be a combination of database queries and program scripts to extract the data into a logical model of the legacy application, plus a further set of queries and scripts to test the data. These can be written by programmers who are familiar with the legacy database. A data migration tool may be used to automate this development.

The writing of the data extraction scripts is unlikely to take very long, but the testing of these scripts is likely to uncover further problems with the quality of the source data. It may therefore require a large number of iterations of the testing and refinement of the scripts and / or the data before this phase is completed. Around 20% of the total project time should be allowed for this phase.

6.5. Data Transform and Load

Once the extract scripts and source data have been tested against the design specification the data can be transformed and loaded into a target database, which should be a copy of the new application database which has been configured in its go-live state. The supplier of the new application may provide routines and support to help achieve this. This exercise may throw up problems with the format or integrity of the data which were overlooked at the design phase. In particular if the data is being obtained from a number of separate sources problems with duplicate or mismatched data may only be identified at this stage. The exercise may also bring to light problems with matching the migrated data to the database configuration. This process can be carried out by a database analyst. Around 20% of the total project time should be allowed for identifying and correcting problems with the data or configuration.

6.6. Acceptance Testing

When the data is successfully loaded into the new application it should undergo a period of acceptance testing by the users. Acceptance test plans should be written by a business analyst and / or experienced users of the existing applications. Acceptance testing will require the involvement of user representatives from each of the teams and job roles that will be using the new application. It will normally take place at the same time as the user acceptance testing of the functionality of the new application, but it can be done as a separate exercise once the application has been tested with ‘new’ (i.e. manually entered) data. At least 3 iterations of acceptance testing should be allowed; the first should identify any high level issues, the second should identify lower level problems that may have been overlooked during the first iteration, and the third should confirm that all the issues have been resolved. In total this period should account for around 20% of the total project time.

6.7. Go Live

Finally the steps required for the switch from the current applications to the new application should be planned. A period of time will need to be allowed between the time the old applications are ‘switched-off’ (i.e. no longer used for input) and the time that users can start to use the new application in the live environment. This time will typically range from a couple of days to a week. During this period of time the ‘final cut’ of the current data needs to be extracted, checked and loaded into the new application. There are also likely to be a number of operations to be carried out on the migrated data before it can be used – such as setting up access rights for the system users. Consideration will need to given as to how any data collected in this period will be retrospectively entered into the new application and who will do this task.

When deciding on the go-live date for the new application consideration should be given to the potential issues around reporting on migrated and retrospectively entered data. This data will not necessarily conform to the rules that apply to ‘live’ data, either because of the difficulties in translating data between the applications or because the start and end dates of pieces of work cannot be correctly recorded. Hence reports that assume these rules do apply may not give the expected results. If possible the go-live date should be set to the start of a reporting period. If this is not possible then it may be necessary to enter some data in both the existing and new applications in order to produce consistent reports.

7 Specification

It is recommended that two separate specifications are written as there are two target audiences that need to be informed of how the data will be migrated. The first audience are the users of the existing and new applications. These people will need to be told how the information that they have in their current applications will appear in the new application. The second audience are the technical staff who will be extracting the data from the existing databases. These people will need to be told where the information that is required for the new application is to be taken from and what format it should be exported in.

7.1. Requirements Specification

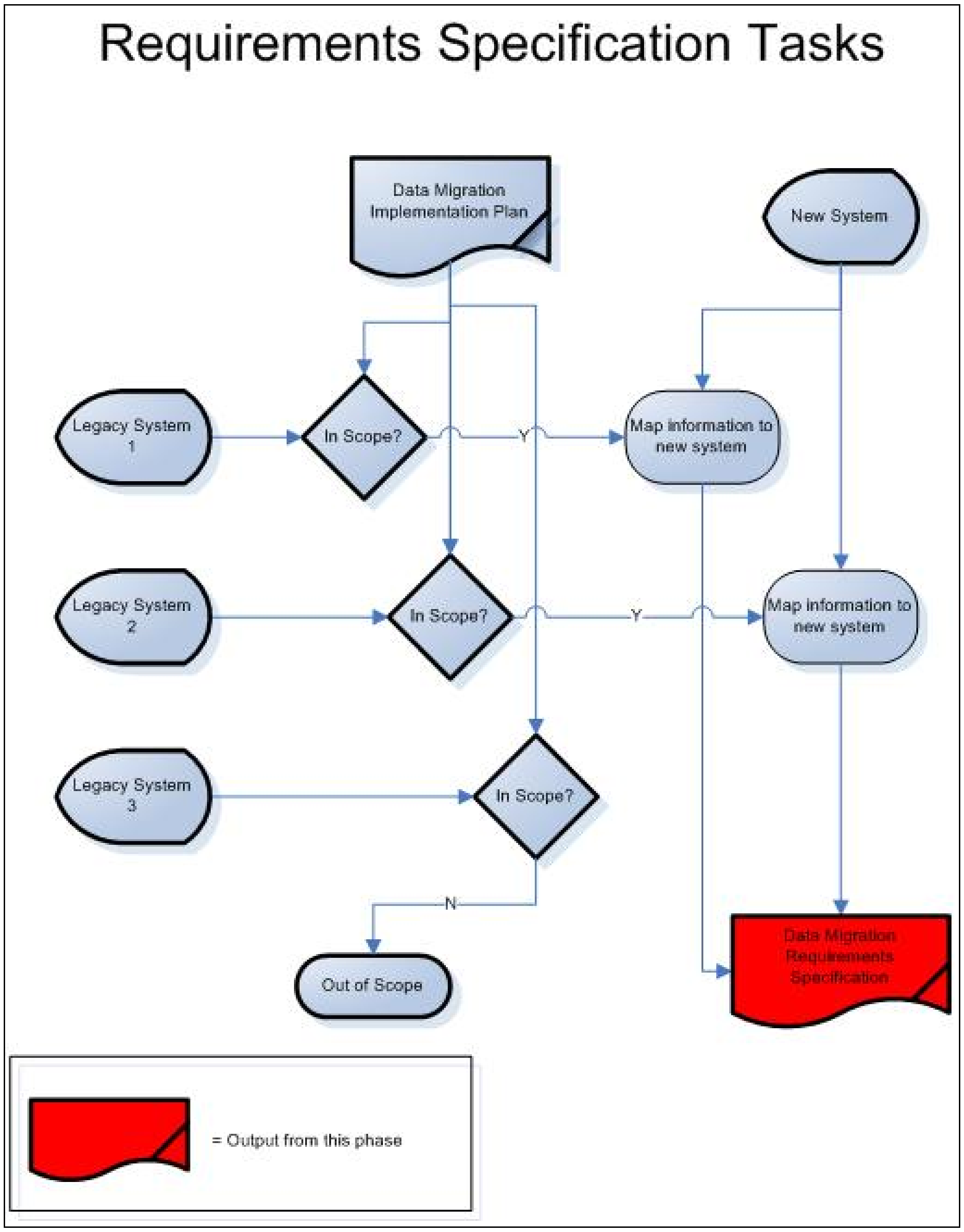
Requirements Specification Tasks
Data Migration Implementation Plan
New System
Legacy System 1
In Scope?
Y
Map information to new system
Legacy System 2
In Scope?
Y
Map information to new system
Legacy System 3
In Scope?
N
Out of Scope
Data Migration Requirements Specification
= Output from this phase

Figure 4: Requirements Specification Tasks

This specification is written for the application users. Its purpose is to inform the users of the information that is, and isn't, being migrated and obtain their agreement for this. It should be written from the perspective of a user of the current application (i.e. the source of the information). One possible format for the specification is to take each of the screens in the current application and show – with screenshots if possible – where the data that currently appears on that screen will appear in the new application.

Specification

The specification should explain both what data is being migrated and where it is being migrated to. The decision as to what data will be migrated from a particular application will be driven by the same considerations as were used to decide which applications should be included in the migration at the scoping phase.

To arrive at this decision the business analyst should interview the users of the current application, including the people who may use it to produce statutory reports and management information.

It may be decided that only a subset of a particular type of information in the current application will be migrated. For example if the existing application contains many clients who are no longer being provided with services then a decision could be taken to only migrate those clients who have been provided with a service within the last five years. The justification for doing this will be to reduce the amount of data cleansing required and to ensure that data held in the new application conforms to data protection guidelines regarding retention of client data.

The decision as to what data will be migrated will also be influenced by where it will appear in the new application. If there is no screen or field that directly corresponds to that in the existing application then the data may be able to be put into a ‘notes’ field or into a user configurable screen or field within the new application. Alternatively it may be able to be attached as a part of a document or it may be left out of the migration completely. Where complex data could be migrated in a number of different ways it may be necessary to carry out a feasibility study of the different options involving trial migrations of sample data.

If there is some data in the current application which cannot or should not be migrated to the new application then an option is to extract it to an archive of some sort. This option should only be used if it will not be practical for users to access the old applications after the new application goes live as the establishment of a separate data archive will involve extra development effort.

7.2. Design Specification

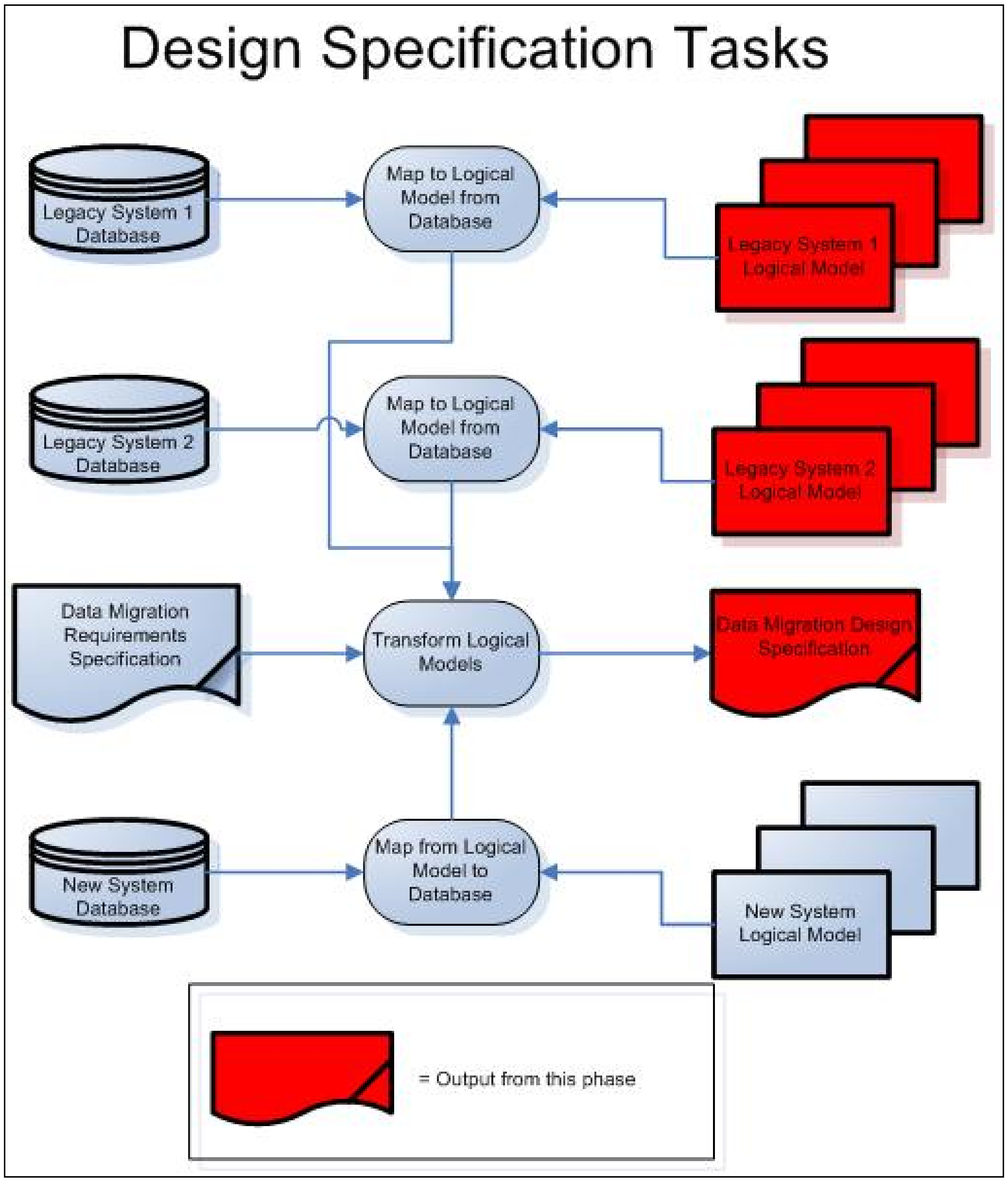
Design Specification Tasks
Legacy System 1 Database
Map to Logical Model from Database
Legacy System 1 Logical Model
Legacy System 2 Database
Map to Logical Model from Database
Legacy System 2 Logical Model
Data Migration Requirements Specification
Transform Logical Models
Data Migration Design Specification
New System Database
Map from Logical Model to Database
New System Logical Model
= Output from this phase

Figure 5: Design Specification Tasks

This specification is written for the technical staff who will be extracting the data and checking the data quality. The specification will take the decisions made in the requirements specification and translate them into the operations that will enable the data in the source database to be extracted, transformed and loaded into the target database.

It can sometimes be difficult to understand how information is held in a database purely by examining the tables in the database. These tables will have been designed to support the physical requirements of the application for storing and retrieving data, which are not always the same as for a human being to process that information. It is therefore recommended that logical models are created of the information to be migrated, much in the same way as management information reporting tools will present the information for users to query or write reports from. A logical model should be created for each of the source applications and for the target application.

These logical models can be created using applications that support data modelling such as Microsoft Visio. If you are using a tool for data migration it may well have a visual interface that will allow you to create the models directly in the tool. In this case the design specification can include diagrams of the models taken from the migration tool.

The design specification should also describe how the physical data from the source databases will be extracted into the logical models of those databases, how the data in the logical models of the source databases will be transformed into the logical model of the target database, and finally how the data from the logical model of the target database will be loaded into that database. Each of these should be presented from the perspective of the receiving environment, i.e. the extract should show how the logical model of the source database is populated from the physical database, the transformation should show how the logical model of the target database is populated from the logical model of the source database, and the load should show how the physical target database is populated from its logical model. The specification should provide enough information for the users of the tools to create the migration scripts and to be able to test that they are meeting the requirements.

The logical models of the source databases need only contain the scope of information that has been included in the requirements specification. For example if the old application contains health information about a client but the new application does not support the maintenance of this information then the health information should not be part of the logical model. To include this would be a waste of effort because that information will not be part of the transformation.

Where the requirements specify that a subset of a particular type of information should be migrated (e.g. clients who have received a service in the current year) this selection can either be done in the transformation process or in the extract process. The advantage of carrying out the selection in the extract process is that it will reduce the number of records that need to be manipulated and avoid the need for the transformation process to have to deal with records that are not required in the source database. For example if the requirements state that only current records need to be migrated then obsolete types of information will not need to be dealt with in the transformation. The disadvantage of carrying out the selection in the extract process is that it can complicate the validation required to ensure that the correct number of records have been extracted, especially if the criteria are complex and do not relate directly to data held in the source database.

The logical models should not contain codes for information that are dependent upon the set up or configuration of the physical database. The models should hold information in the manner in which it is presented to the user. This will allow the transformation to be determined from the requirements specification. The translation of codes to descriptions and vice versa should be done as part of the extract and load mappings.

8 Implementation

8.1. Development Environment

You will need an environment in which to develop and run the processes to extract, transform and load the migrated data.

This environment should have the following:

- connections to the database containing the data to be extracted;
- a means of creating the logical models of the source and target databases;
- a means of manipulating data;
- connections to the database in to which the data will be loaded.

A simple tool that meets these requirements is Microsoft Access. You can use the ODBC facilities within Access to connect to external databases of most types.

If both your source and target databases run on the same platform then you can use specialist client development tools for that platform, such as Management Studio for Microsoft SQL Server or SQL Developer for Oracle.

There are also specialised data migration tools available such as:

- Trillium
- Centerprise Data Integrator
- Talend Open Studio

8.2. Data Extraction and Cleansing

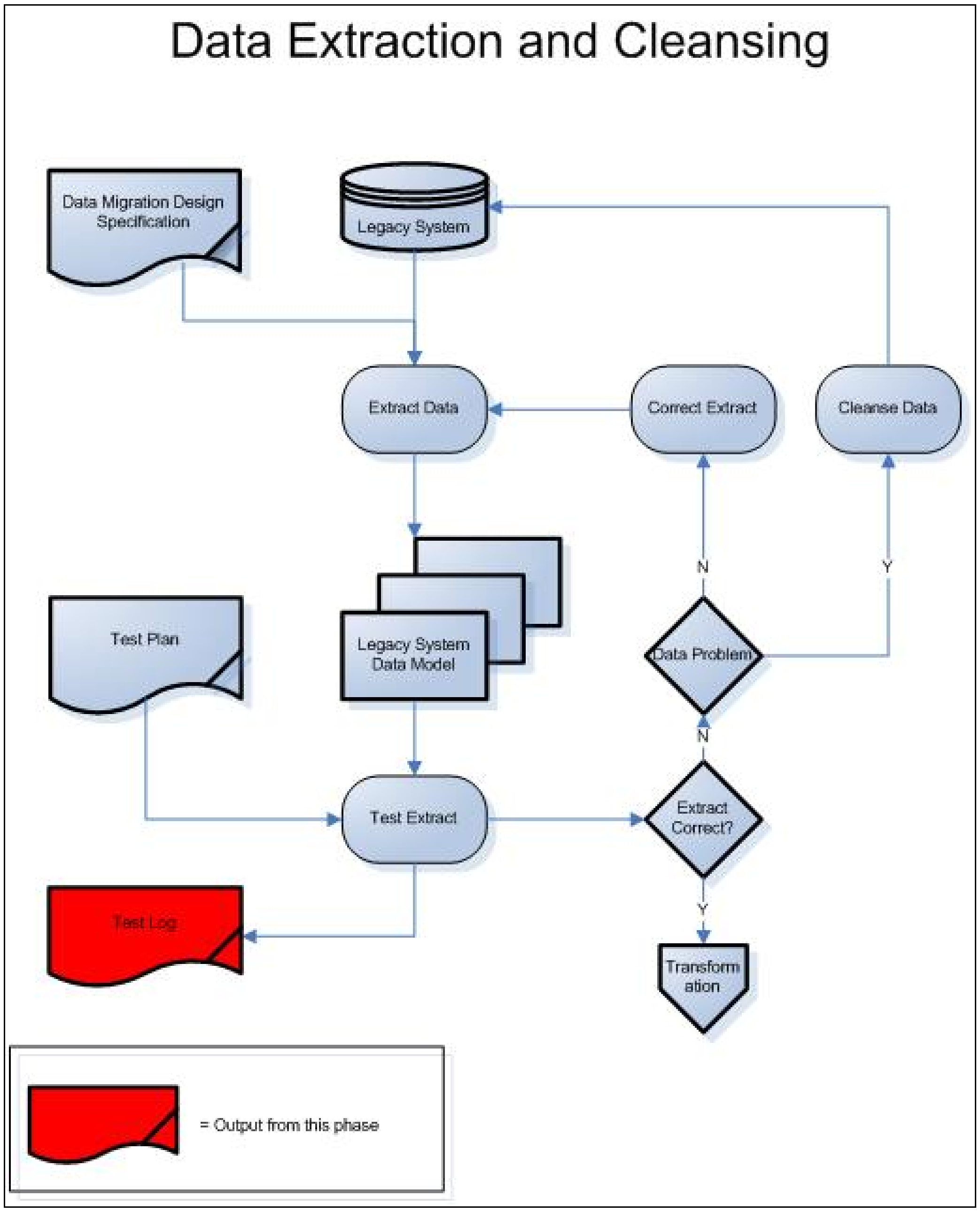

Figure 6: Data Extraction & Cleansing Tasks

Data extraction is the process of retrieving the data from the source database into a logical model of the source application. This is usually done by writing SQL Select queries on the source database tables or using a migration tool that allows you to specify what data should be extracted.

The data extract process should ensure that the data held in the logical model obeys the business rules of the source application. This should be done where possible by applying constraints to the logical model. Where this is not possible the business rules can be checked for in the extract scripts.

Where the data does not meet the business rules criteria it will need to be cleansed. Data cleansing comprises the following main operations described in the sections below:

- Domain checking
- Enforcing integrity constraints
- De-duplication

Data cleansing can either be carried out directly in the source database, in the data migration scripts, or by manipulating the extracted data. The pros and cons of each approach are listed in the table below.

Cleansing Approach	**Pros**	**Cons**
In source database	Only needs to be done once Allows users to determine correct values in each individual case	May not be able to be done in source application Requires co-operation of users
In data migration scripts	Is done automatically Does not affect source application Does not rely on user co-operation	Does not allow for consideration of individual cases
Manipulating extracted data	Does not affect source application Allows for consideration of individual cases Does not rely on user co-operation	Has to be done on each data extract Relies on knowledge of extracted data which only users may possess Prone to errors

Table 1: Comparison of Data Cleansing Approaches

8.2.1. Domain Checking

Domain checking involves checking that the values in a particular field or column in a database fall within the range of allowed values. Ideally constraints should be placed on the columns to ensure this but this may not have been done, either because the database designer was not aware of the limitations on the values or because it was not possible to do this in the DBMS.

Examples of common domain checks are given below:

- Checking that a text field that is used to indicate whether or not a record has a property is ‘Y’ or ‘N’;
- Checking that a numeric field that is used for the same purpose is 0 or 1;
- Checking that a text field that is used to indicate the state of a record only has the allowed state values;
- Checking that a date field only contains a date range that is valid within the application, e.g. a field used for date of birth should not be more than 120 years in the past or a future date;
- Checking that an account reference, post code or telephone number has a valid combination of numeric and alphabetic characters.

8.2.2. Enforcing Integrity Constraints

Enforcing integrity constraints involves checking values which should match across different records or rows in the database. This can often be enforced within the DBMS but again there may be reasons why the database designer was unable to do this. Some examples are:

- Checking that a client identifier in a transaction record corresponds to an identifier of a client record;
- Checking that an invoice header record has at least one invoice line associated with it;
- Checking that a total figure in an invoice header record matches the sum of the corresponding figures in the records which are part of that invoice;
- Checking that a date for an account transaction falls within the range of dates for which the account was open.

8.2.3. De-Duplication

De-duplication involves checking for and removing records where more than one record or row in a database has been used to hold information for the same entity.

In many database tables there is no natural primary key which can be used to uniquely identify an entity; for example a person may be given a unique identifier within an application, but another user may not realise that the person has already been recorded and create a second record with a different unique identifier for the same person.

Checks should therefore be made on the key fields within a record to identify where there are duplicate values for those fields across more than one record, and only one of the duplicate records should be extracted. In some cases it may be valid to have duplicate values – for example in a database containing a large number of people there may be some with the same name and date of birth – so some manual checking may be required before records are deleted.

Where the records contain identical information it is straightforward to just extract a single record by specifying that only distinct records should be extracted; however if some of the information is different then this information may need to be merged.

8.3. Data Transformation

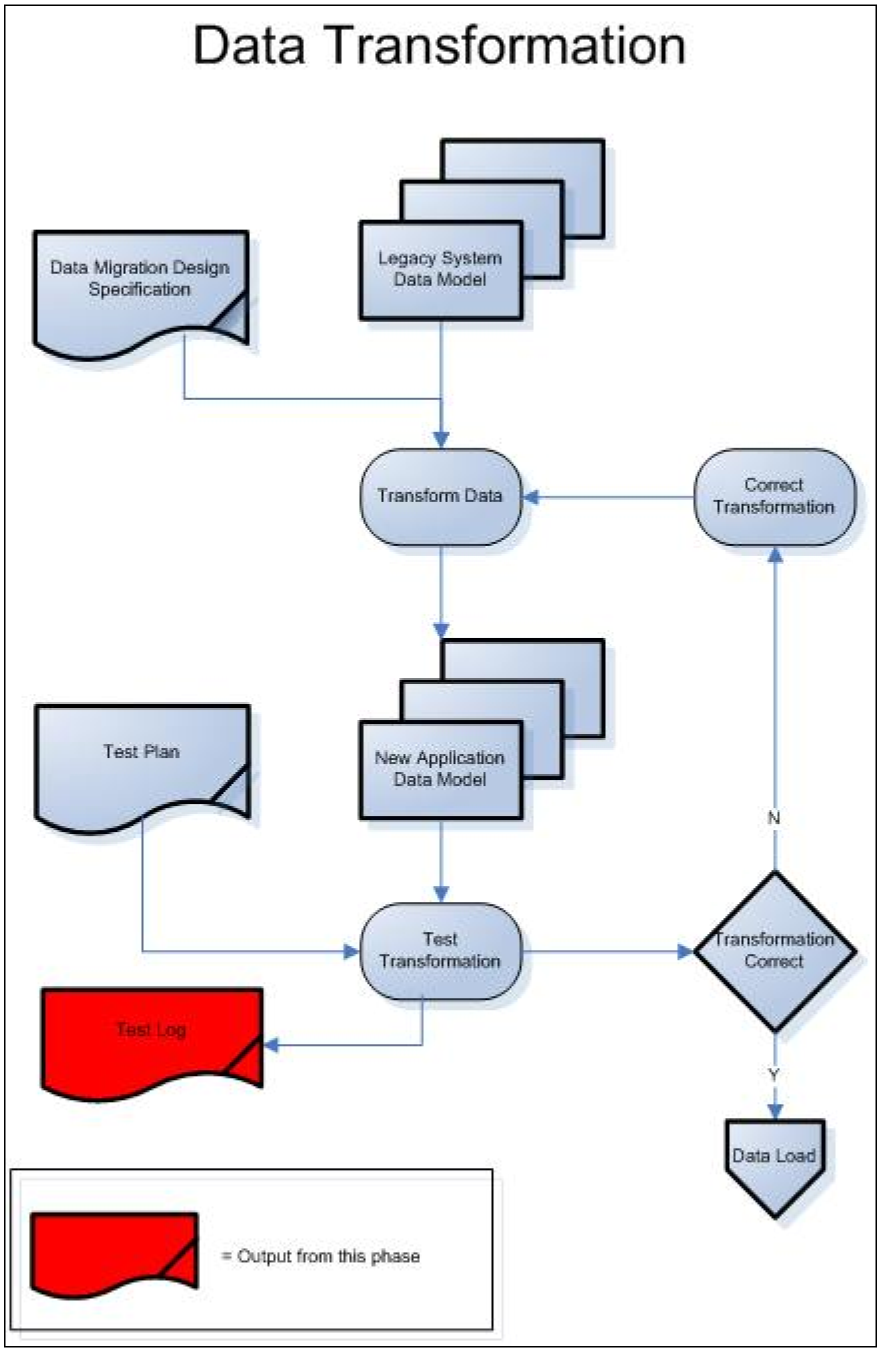

Figure 7: Data Transformation Tasks

Data transformation is the process of transforming the data in the logical models of the source applications into the logical model of the target application.

During this process the data in the source model must be manipulated into the target model whilst ensuring that the business rules of the target application are adhered to and as little information is lost as possible. Remember that the transformation process should only carry out operations specified in the data migration requirements. Data that is hidden from the user should only be used or created during the extract and load processes.

The main operations used in the transformation process are mapping, filtering, separating and combining.

8.3.1. Mapping

The contents of a table column in the target model will usually be taken from a corresponding column in the source model. Where the data is mandatory in the target model but there is no corresponding data in the source model a default value should be used. For example the target model may require a start date for an address where none exists in the source model. In this case either the current date or a ‘dummy’ date such as 1st January 1900 could be used.

Where possible you should use the same column names and domains in the logical models of both the source and target applications where you intend the same information to be held in each. For example if both applications hold information about a person’s ethnicity then call the column where this information is held in each model ‘Ethnicity’ and allow the same range of values in each. If the coding of ethnicity is different in the source and target applications then translate the values during the extract and/or load phases.

8.3.2. Filtering

The migration requirements may specify that only a subset of the information in the existing application is required in the new application. In this case filters (or SQL ‘where’ clauses) should be applied to the source model to restrict the data copied to the target model. For example information that is more than 5 years old may not be required in the new application, and so a filter which checks a column holding the date of the information against the current date will be required. Ensure that the criteria that determines how the data is filtered is specified in terms of the logical model so that it can be verified during user acceptance testing.

8.3.3. Separating

Information held in one place in the source model may be held in different places in the source model. In this case the data from a single row or column in the source model should be separated into multiple rows or columns in the target model. For example a person’s address may have been stored with the person’s details in the source model, whereas the target model will allow multiple addresses for a person and so stores addresses in a different table.

8.3.4. Combining

Conversely information held in separate places in the source model, or in more than one source model, may need to be combined into a single place in the target model. For example the services being delivered to a client may currently be held in different service type tables or databases but should be combined into a single service table in the target model.

Where data needs to be combined from different source models extra effort will be required to ensure that the data matches, i.e. a common means of identifying who or what the information belongs to on each application will need to be found. This often involves the manual addition of reference numbers to one or other of the source applications with the attendant risks that involves. For that reason combining data from different models should be avoided, and where it is essential extra time should be allowed when estimating the resource requirements for the implementation and testing phases.

8.4. Data Loading

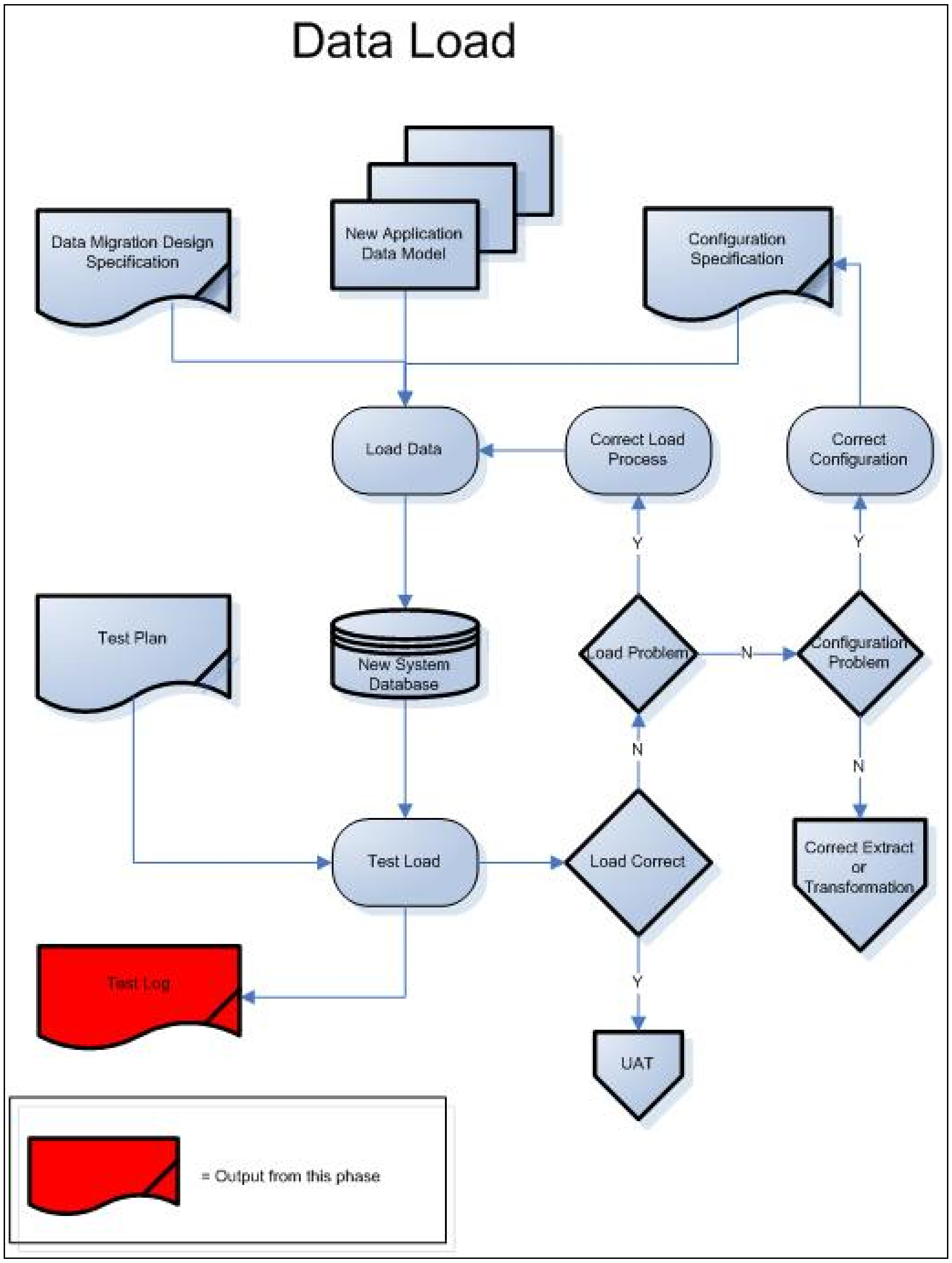

Figure 8: Data Load Tasks

Data loading is the process of moving the information in the logical model of the target application into the physical database tables. This is usually achieved by writing SQL Insert or Update scripts.

To enable this to be done successfully you must have a thorough understanding of how the application stores data in the physical database. Where possible you should use tools or documentation developed by the manufacturers of the target database application. If the application uses an API (Application Program Interface) to store data in the database then you should use this too, except in cases where this significantly slows down the performance of the data load.

Where these tools and documentation are not available you should experiment with entering data into the application through the user interface to determine how that data is stored in the physical database. Note that this exercise can also be helpful in determining the business rules of the application.

The data load scripts will map rows and columns in the logical database to the physical database and translate logical values into the physical codes used to represent them. Because different copies of a database could use different codes to represent the same logical value the data load scripts should use look-up tables to translate logical values to physical codes. These tables should be checked against the configuration before loading. Including a description of the information being translated will help identify any problems with mismatched codes.

8.5. Testing

8.5.1. Test Plans

All data testing should be planned. A test plan should be created for each set of tests to be done which lists the tests that need to be carried out on that data.

The test results can either be recorded on the test plan or on a separate test log. The test results should include:

- The purpose of the test
- The date and time it took place
- Who carried out the test
- What the expected results of the test were
- What the actual results of the test were

8.5.2. Data Extract Testing

The purpose of data extract testing is to ensure that:

a) The correct number of records are extracted into the logical data model of the source application;

b) All the records in the logical data model conform to the business rules of that model.

The criteria specifying the records that should be extracted into the logical data model should be simple enough to allow reports that already exist (or are simple to create) within the source application to be used to verify the number of records extracted.

The business rules of the logical model can be used to determine the number of records that should be extracted; for example if the rules state that each person should have one and only one address then the number of address records extracted should equal the number of person records extracted.

Discrepancies between the expected and actual number of records extracted will usually indicate that data cleansing needs to be done on the source data; if not then the extract criteria are probably too complicated. The testing to ensure that the extracted data conforms to the business rules of the application will verify that the data cleansing has been done successfully.

8.5.3. Data Transformation Testing

The purpose of data transformation testing is to ensure that:

a) The correct number of records are inserted into the logical data model of the target application;

b) All the records in the logical data model conform to the business rules of that model;

c) The field level transformations between the target and source models have been carried out according to the requirements specification.

The criteria in the requirements specification for filtering, separating and combining records should allow simple queries to be written comparing the numbers of records in the source and target logical models.

The business rules can often be checked by applying constraints to the target logical data model. Records that do not conform to the rules should be trapped by error handling routines with the transformation process and will show up as discrepancies between the number of records that should be in the data model and the number that are.

Where this is not possible representative samples of the data should be checked manually or by a simple script. If a rule states that a record should have one value or another value then a good check is that there are no records with neither value, i.e. if A=X or A=Y must be true then check that A <> X and A<> Y is always false.

Sometimes counting the records in the logical model can identify when a business rule has been broken. For example if each person should have one and only one main address then the number of main addresses in the database should equal the number of people.

Field level transformations will usually need to be checked manually by comparing representative samples from the source and target models.

8.5.4. Data Load Testing

The purpose of data load testing is to ensure that:

a) The correct number of records are written into the physical database of the target application;

b) The data written matches the configuration of the physical database;

c) The data written to the physical database can be correctly handled by the application.

The design specification should state how the records in the logical model of the database are written to the physical model and the data load testing will be used to verify that this design has been correctly implemented by comparing the actual numbers of records in the physical database to the expected numbers. Discrepancies in these numbers will imply either that the implementation has not interpreted the design correctly, or that records have been rejected by the physical database because it has not been configured as expected. For example if you are migrating cost codes associated with a purchase of a service the configuration of the new application may not have been set up to allow those codes. This could indicate that the configuration is invalid, that your cost code transformation is invalid, or that invalid information exists in the source database.

The final possibility given above is particularly difficult to deal with. You could build the check into the business rules for the logical data model of the source application, however the information may conform to the rules of the application itself (which may not check cost codes) in spite of being invalid in the 'real world'. Alternatively you could deal with it in the transformation process if the rule it falls foul of can be stated independently of the configuration of the new application. Note that if you assume that the configuration is correct then you should not deal with the problem as part of the load process because the information itself is wrong, not just the physical data it is being translated into.

A test script should be written to verify that the user interface and functionality of the application works correctly with migrated data. This test script should be based on data held in the logical model – which should be a representation of how that information is presented in the application. Therefore the assumption should be that data stored in the logical model will behave in the same way in the application as data entered through the user interface of the application. Note that if you are using a API provided by the supplier of the application it should only be necessary to carry out application testing to demonstrate that you are using the API correctly (and that it has been written correctly!).

8.5.5. User Acceptance Testing

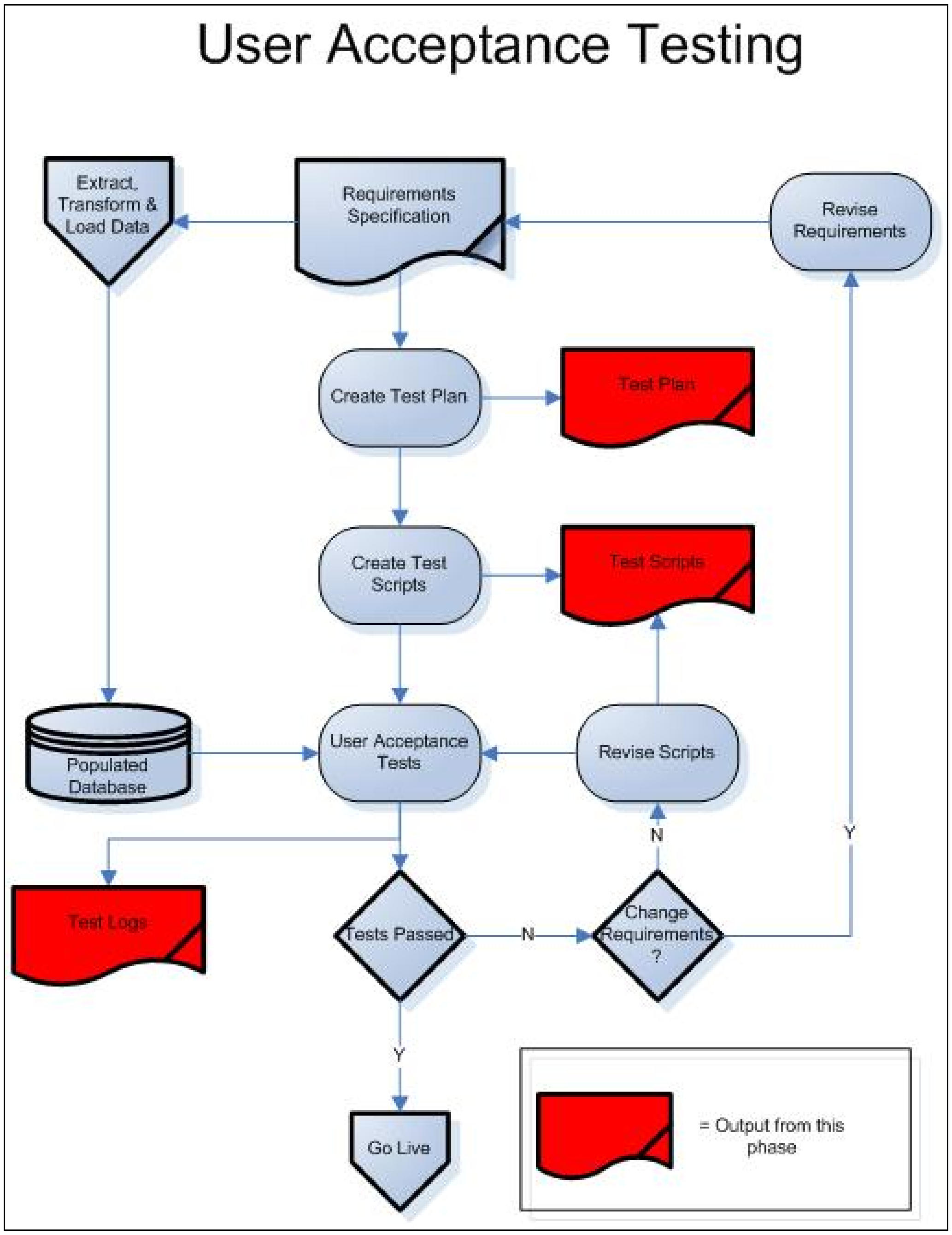

Figure 9: User Acceptance Testing Tasks

The tests described in the previous sections are designed to confirm that the design specification has been implemented correctly and that it has correctly interpreted the requirements specification.

User acceptance testing is used to confirm that the requirements specification has correctly identified the needs of the users for migrated data to support their business processes.

Acceptance testing is best achieved by writing a set of scripts that can be followed by the users of the application. For this reason the acceptance test scripts should not be written by members of the data migration team, they should be written by user representatives who have used the existing database application and who are familiar with the new application.

An acceptance test script should be written for each use case (i.e. function carried out by a particular type of user). The use cases should have been identified during the implementation of the new application itself and should already have been tested with new data entered directly into the application. The writers of the acceptance scripts should be familiar with the decisions made by the project board at the scoping phase of the migration so that they do not test for data that was never intended to be migrated (e.g. if a business decision was made to exclude closed cases then test scripts should not be written which assume that information for closed cases will be available).

If the extract, transform and load tests have been done properly then any problems reported during acceptance testing will indicate either that the requirements specification requires changing or that the test scripts have not interpreted it correctly. In the former case it may be that the requirements specification overlooked particular issues or has interpreted users' needs incorrectly. The project manager should make a judgement on the impact of changing the requirements specification in terms of the time and resource required to alter the design specification, migration scripts and test plan weighed against the problems reported. If it would be necessary to put back the go live date then the impact on the business of doing this needs to be measured against impact of leaving the migrated data as it is. In some cases it may be possible to put a workaround in place and / or alter the migrated data after go live.

If it is decided that the requirements specification does not need to change then the test scripts should be altered to take into account the issues raised so that there is a record that the problems identified have been considered.

8.6. Go Live

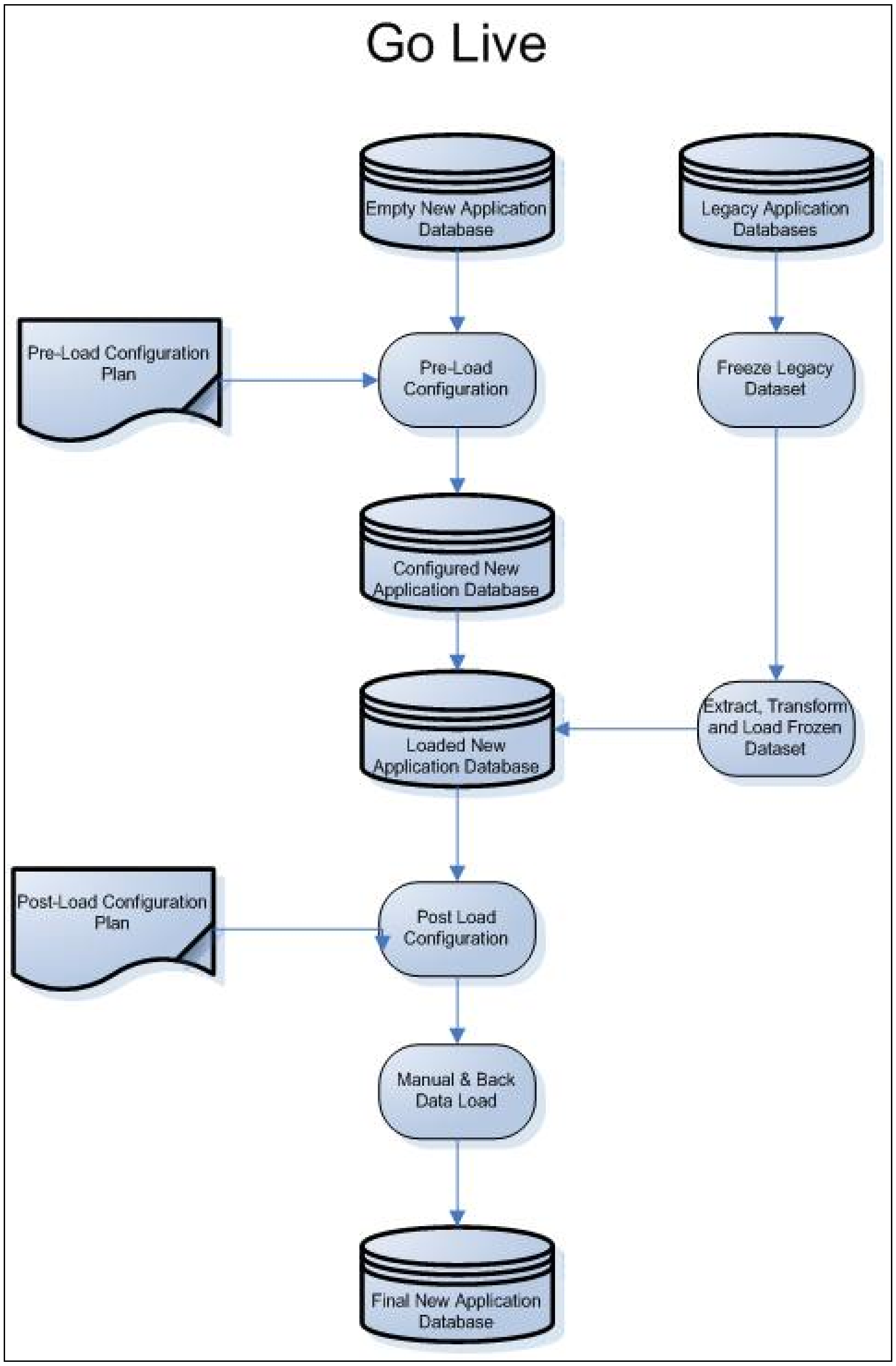
Go Live
Empty New Application Database
Legacy Application Databases
Pre-Load Configuration Plan
Pre-Load Configuration
Freeze Legacy Dataset
Configured New Application Database
Loaded New Application Database
Extract, Transform and Load Frozen Dataset
Post-Load Configuration Plan
Post Load Configuration
Manual & Back Data Load
Final New Application Database

Figure 10: Go Live Tasks

Once the migrated data has passed the acceptance tests plans can be made for the final data migration which will populate the live database. The following steps should be included in this plan:

- ‘Freezing’ the applications from which the data is being migrated
- Extracting and checking the data from the old applications
- Transforming the extracted data
- Configuring the live database
- Loading the migrated data into the configured database
- Carrying out any application configuration that relies upon the migrated data
- Manually entering data that is not being migrated
- Back-loading data that has arrived after the old applications were frozen

Users will need to be informed well in advance as to when their existing applications will be ‘frozen’. This is the point at which the final data extract will be taken from those applications. It does not necessarily mean that they will no longer be able to use those applications, but any data that is entered from this point onwards will not be migrated and will have to be manually entered into the new application. A copy must be taken of all the old application data at the point at which the application was frozen and the data extracts should take place from this copy. (An alternative would be to make the ‘live’ copy read only and allow data entry into a copy of the application data).

Although the data extract scripts will have been tested it is possible that bad or unexpected data could have been entered into the old applications since the testing was carried out. It is therefore necessary to check the final extract and allow some time for possible corrections to be made to it. Note that the corrections must be made to the frozen copy of the data that was used to take the data extract from.

If the live application database is not the same as that which was used for migration testing then the data load files will need to be altered so that they load the data to the live database. If this is the case a dress rehearsal of this process should be carried out prior to the final data load to uncover any potential problems. An alternative approach is to perform the go-live migration into the same migration database as has been used for testing and then copy that to the live database as the final step of the process. The latter approach is recommended when migrating data into a database that is already in use (e.g. during phase 2 of an implementation).

If there are any configuration tasks that rely on migrated data, e.g. specifying security rights for application users where the user details are migrated from an existing application, these tasks will have to be carried our after the data has been loaded but before the go live date. It may also have been decided that is easier to load some data manually than to create extract files for it, and this data may also rely upon migrated data being present.

The total amount of time required for all these go live actions may be more than is available in non-working time (e.g. over a weekend), and so consideration may need to be given as to how to enter the data into the new application that has arrived between the date on which the old applications were frozen and the date at which manual input into the new application can commence.

---

[i] Infomig Data Migration Implementation Plan (MS Word document)

[ii] Generic Infomig Data Migration Project Plan (MS Project document)