

TESTING OF MIGRATIONS BASED ON TMAP®

Author	Rob Baarda
Version	1.0
Location	Diemen
Document	August 2005

VERSION INFORMATION

Version	Date	Remarks	Author
1.0	August 2005	Dutch paper	Rob Baarda
1.1	May 2006	English translation	Gina Koobs

Copyright © Sogeti Netherlands BV, Diemen. All rights reserved.

All rights reserved, Barring any statutory exceptions, no part of this publication may be reproduced and/or published, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical photocopying, recording, microfilm or otherwise without the written permission of the rightful claimant(s) of the copyright or, as the case may be, the publisher of this publication. This copyright also applies to full or partial editing of this publication.

CONTENTS

VERSION INFORMATION	II
CONTENTS	III
1 INTRODUCTION	1
2 MIGRATIONS.....	3
2.1 Migration types	3
2.2 Possible migration targets	4
2.3 Migration project approach & life-cycle	4
2.4 Organisation aspects of a migration project.....	5
3 SPECIFIC POINTS OF ATTENTION FOR MIGRATION	6
3.1 Which quality attributes of the application could be influenced by the migration? ..	6
3.2 Which quality attributes play a role in migration itself?	7
4 TEST APPROACH OF MIGRATIONS.....	9
4.1 Why test	9
4.2 Test assignment / Assignment formulation	9
4.3 Test basis.....	11
4.4 Test strategy.....	11
4.4.1 Migration specific clues for establishing product risk analysis	12
4.4.2 Test strategy considerations.....	13
4.4.3 Master test plan en test level	14
4.4.4 Detailed test plan	15
4.5 Test life-cycle	15
4.6 Techniques and test types.....	16
4.6.1 Budget aspects	16
4.6.2 Review technique.....	16
4.6.3 Test specification and execution techniques	17
4.6.4 Test forms	19
4.7 Organisational aspects	19
4.7.1 Organisations involved.....	19
4.7.2 Test- en application expertise	19
4.7.3 Defect procedure	19
4.8 Test infrastructure aspects.....	20
4.8.1 Test environment files.....	20
4.8.2 Tools	20
4.8.3 Office environment	21

5	EXPERIENCES TESTING DURING MIGRATIONS	22
5.1	What do testers find in migration practice?	22
5.2	Experiences with budget and realisation.....	23
5.2.1	<i>Realisation formulating master test plan</i>	23
5.2.2	<i>Plan and/or realisation migration project including testing</i>	23
6	APPENDIX: THE WORLD OF MIGRATION	25
6.1	Program code	25
6.2	Data(base) management system.....	26
6.3	Data conversion/data quality	26
7	CHECKLIST TECHNICAL RISKS.....	27
8	LITERATURE.....	28

Introduction

1 INTRODUCTION

Migration occurs increasingly frequent in IT. Reasons for this often are the possibilities of cost reduction by replacing the mainframe, or the disappearance of older technology. The existing applications often contain valuable and approved functionalities, which are worthwhile to migrate. Because migrations often relate to information systems that are critical for business, the damage is substantial when faults are present in the system after the migration: the risks are large. These risks can be covered preventatively by a quality guaranteeing project approach. In places where insecurities still exist the quality of the migrated information system could be measured.

In this paper the test approach for a successful migration is described.

This approach is based on TMap® [1], the test approach formulated by Sogeti, and the improvements described in TMap Test Topics [3]. Furthermore the Sogeti migration approach Mikado [2] is used.

Besides these documents the experience of the members of the 'SC Testmanagementplatform' and of the participants of the migration testing session in SC Randstad-North are used. The reviewers Norbert Mimpfen, Ralph Klomp en Tim Koomen have contributed as well with their acute remarks and their experiences. Gentlemen, I thank you.

The following manner of specification is used:

Firstly chapter 2 "Migrations" globally describes what migration means, for which the numerous experiences from practice are used as input.

Even though migration is not supposed to change anything apart from that which needs to be changed, several things could be adjusted unintentionally. These are described in chapter 3 "Specific points of interest for migration"

Next the approach for the test process is described in chapter 4 "Test approach of migrations" for which the order of the TMap book [1] is followed extended with the additions from TMap Test Topics [3]. It was chosen to describe the migration specific topics alone.

In the final chapter experiences of testers and examples from practice with regard to the quantitative contribution of testing in a project are represented.

In the appendices solutions for migrations in the Netherlands are sketched as well as a detailed list with points of risk to form a picture.

Target audience

The white paper is meant for migration specialists and experts on testing based on the TMap approach. Migration specialists will possibly need additional explanation of aforementioned test specialists for the test approach.

Scope

The scope of this document is the migration of information systems (applications). Migration of workplaces is not taken into consideration in this version. The quality of data is also left out of consideration; see TMap Test Topics [3], chapter Testing of Data warehouses /Business Intelligence for an onset. The introduction or upgrade of a package is not considered to be migration; for this subject the white papers "Testing en QA of packages" and "Testing of the upgrade of SAP" are available. The testing of the infrastructure and the system software is outside the scope of this paper.

Introduction

Concepts

Specific migration concepts are:

Proof of Concept, also known as Proof of Confidence (POC).

In the start-up phase of the migration project it is reviewed if the migration will not experience problems by migrating the typical parts on a small scale. If the quality of the data (base) is unknown (part of) the data conversion in the POC is often considered as well.

Pre-migration situation and post-migration situation

Pre: the situation before migration, post: the situation after the migration

Migration = Conversion?

IT-definitions of the concepts migration and conversion have not been found
Migrating is often one abstraction level higher, for example the programs and/or the database is converted and the system migrated. This line of thought is followed in this document as well.

Migrations**2 MIGRATIONS**

This chapter describes some important characterisations of migration and migration projects.

2.1 Migration types

A migration starts with (wanting/having to) change components in and/or around an information system (see column 1 in table below for possible situations). Changing these components often has consequences for other components. These need to be migrated to the new situation. See the 2nd column in the table below for possible consequences.

What can change	What to migrate	Example
Programming language	Software (including screens). Sometimes data	From Bull-Cobol to MicrofocusCobol From Cobol to C
Version of programming language	Software	From Oracle 5i to Oracle 6i
Data base system or version of data base system	Software & data; Related (control) tools	From IDMS to DB2 From Oracle 5i to Oracle9
System software	"JCL", job scheduling etc.	From VAX-DCL to IBM-JCL
Version system software	Software	From CICS x to CICS x+1
Hardware/infrastructure (excluding desktop)	Usually everything on top of this: JCL, scheduling, database system, data, software (including screens)	From IBM S/390 to AS/400 Server W-NT to W2000
Functional	Software Data	Y2000, Euro, extending the number in digits in for instance telephone numbers or bank accountnumbers
Combine departments/organisations	Data Software	Combine two (approximately) as for structure equal databases Often some functional changes
Introduction or replacement of software package	Data Software	Introduce SAP, CRM Migration of web content to CMS
Version of software package	Self built software	From SAP 4.1 to SAP 4.6
Desktop including OS	Work place	From Windows-NT to Windows-XP

NB The final 3 kinds of migration (between the double line) are not discussed in the document.

Besides the here mentioned component types some extra components can sometimes be migrated/converted: functional documentation, technical documentation, testware (usually for regression tests).

Migrations

2.2 Possible migration targets

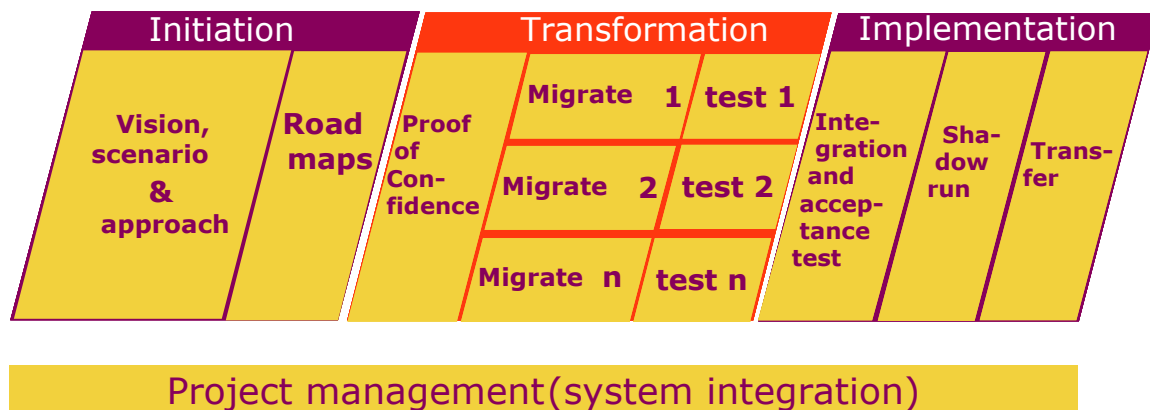
The goal for most migrations is that the functionality keeps working 1:1 in the new technical environment.

An additional goal could be setting a proper basis for the future by, for example:

- Removing the not used code (=dead code) from the programs
- Restructuring program code (remove trickeries)
- Not migrating not-used programs
- Enriching the data during or before the data migration (for example to complete or to structurally solve misuse of data fields)

Above mentioned goals are of a functional character. Other properties of the information system need to remain more or less the same. Chapter 3.1 "Which quality attributes of the application could be influenced by the migration?" describes this in more detail.

2.3 Migration project approach & life-cycle



In the figure above the main activities on generic level are represented. These will be described shortly below and are explained further in chapter 4 "Test approach of migrations".

Vision, scenario & approach

- Exploration/analysis of object to be migrated
- Detailed analysis to be able to estimate the size, distinguish between the different kinds of components (computer languages, screen dispatch, report generator, JCL/scripts, scheduling, etc) and find potential problem areas.

Road maps

- Make the conversion approach concrete, for example by introducing a big-bang or incremental migration approach
- Inventory of available testware

Proof of Concept (Confidence)

To make sure the "real" migration will succeed a proof-of-concept (POC) is often carried out. This will often contain the complete (trial) conversion of the data.

- For a data conversion, often control mechanisms are built in and special data conversion software is built and tested.
- Data conversion of current production data (trial conversion)
- Perform a Proof of Concept conversion for a selection of software and test this conversion. The selection of the software contains all possible software statements and program structures.
- If desired adjust conversion approach until all structural problems are solved.

Migrations

Migrate and test

- Divide the source material to be converted into clusters.
- Convert programs etc per cluster in a work package

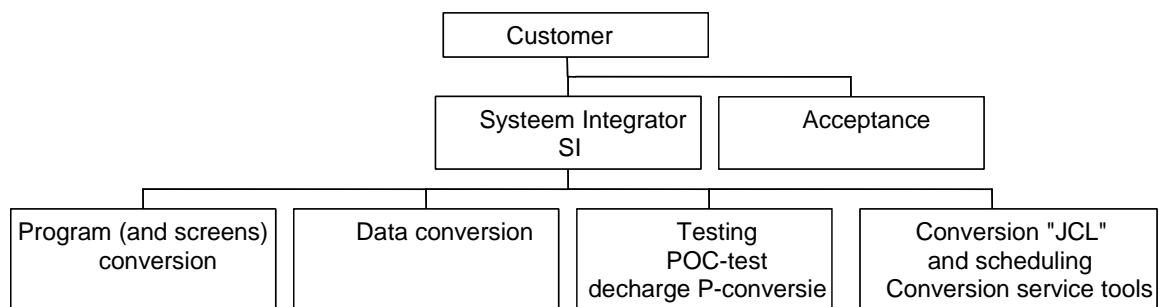
Integrate and accept

- Integration of all software in one environment and testing the integration
- Trial conversion of the data using the conversion plan
- Acceptance by the customer, if necessary by executing tests
- Shadow run
- Transfer to production

For the migration itself automated conversion tools are often used, sometimes completed by smart use of pre-compilers, see chapter 6 Appendix: the world of migrations, where some approaches are described with the names of the companies that practice these approaches.

2.4 Organisation aspects of a migration project

To create an image of how a migration project is organised in practice and to be able to name the main activities in a migration project an organisation structure of a migration-organisation is shown below as an example.



Short explanation

The customer is the sponsor of the migration to be performed.

The system integrator (SI) is responsible for executing the migration. This party could be (a combination of) the internal IT department, the external main supplier of IT or a specialised company (for example Sogeti).

The results of the SI activities need to be accepted and subsequently discharged. It is judged if the acceptance criteria have been met. An acceptance test is often part of the acceptance process. The acceptance test deals in any case with the functionality, usability and often the performance.

The conversion of programs and JCL could be outsourced to specialised companies that often do the job supported by tooling and offshore human resources.

The staff consists of migration specialists, application specialists (developers and system managers), the business process experts and users. Because the migration is often extra work for employees of the migrating organisation, extra manpower is hired amongst whom testers and test management.

3 SPECIFIC POINTS OF ATTENTION FOR MIGRATION

In a migration it is often the intention to transfer the functionality 1:1. The migration of the program conversion needed for the system is often executed in an automated way. Can the system be taken into production without problems or do certain points need attention? This chapter describes the points of attention experienced in practice. In every migration project it needs to be determined which points of attention apply to that specific situation.

The possible influences on the quality attributes have been mapped (paragraph 3.1).

The migration itself also has a number of quality attributes. (Paragraph 3.2).

The used quality attributes are described in more detail in TMap [1, H11].

3.1 Which quality attributes of the application could be influenced by the migration?

1. Functionality: correctness and completeness

Correctness

In a new release an old bug could be solved causing the programmed workaround to lead to faulty processing. To find this out the release notes of the supplier need to be read through carefully.

Relations (interfaces) with other applications could be influenced by the conversion.

N.B. Existing "faults" in the pre-situation will not be solved but brought along to the post situation unless it is agreed that it will be repaired immediately.

Completeness

The functionality has to be completely migrated after the migration.

If the functional processing consists of a number of steps that are built technically with different languages (for example Cobol and C++), there is a chance that this connection does not function anymore after the conversion, so the completeness of the transaction can no longer be guaranteed.

If the system has a different transaction-commit-mechanism in the post situation, the danger of incomplete transactions exists.

2. Performance

When a database management system is migrated there is a fairly large chance at a slower performance caused by for example a changed manner of indexing the data, reading backwards in the database, reading with technical key (direct key), other dispatch of duplicate keys (the option 'duplicates allowed' implemented/not implemented), other data structures by which more complex – computer time-consuming- data mapping of pre to post arises.

If it concerns new hardware an initial tuning problem could arise; the buffering of data could be set incorrectly for example and in the worst case the hardware could be unsuitable.

3. Efficiency

The conversion can cause the efficiency of the system to decrease, for example because the converted software treats the database less efficiently or the software in the converted situation uses more CPU cycles. Efficiency certainly plays a role if the billing of the exploitation of the systems happens based on the number of disk-I/O's and/or the number of CPU-cycles.

Besides this the efficiency of the systems is important because a reduced efficiency of the converted system may decrease the functioning of other systems that use the same infrastructure.

Specific points of attention for migration

4. Security/authorisation combinations of migrated software.
In some applications the authorisation of users and/or data is set up using parameters. The use of parameters is often operating system/software specific. A conversion of OS and/or programming language enlarges the chance of authorisation problems.
5. Maintainability
When source code is converted the readability and thereby the maintainability of the converted code could be harmfully influenced. With an automated conversion the number of lines of source code often increase substantially. For some program structures even with factors. Despite of a comparable program structure the source code could become less readable and thereby less maintainable because of the large amount of lines.
For some conversion demands are even made:
 - The source code needs to be structured better.
 - Standards for software etc need to be complied with. For example: no forbidden statements, no forbidden constructions, compliance with the naming conventions.
6. Controllability
On the new goal platform there are often other service management tools (for example for job scheduling) present or they are missing all together. The effectiveness and efficiency of the service management could –certainly immediately after the migration – have decreased enormously.
7. User friendliness
When the screens are converted, the user friendliness could be increased. If the look-and-feel changes, the user needs to be involved in this, certainly if the order on the screens changes.
8. Connectivity
The interfaces with other systems are always a source of concern. After all something changes on the source side (for example from ASCII to EBCDIC) can the other side of the interface cope with that?
9. Reliability
A reason for migration could be the improvement of the reliability by enlarging the availability because the new system is 'down' less often than the old one. This being less 'down' should then be tested.

3.2 Which quality attributes play a role in migration itself?

The quality attributes below are specific for the conversion/migration itself.

1. Functionality: Correctness and completeness
 - Correctness
Mostly related to the employment of conversion rules for program statements and data mapping of pre to post.
 - Completeness
A data conversion may not lead to loss of information.
If there is a fallout list for the data conversion the sum of the data-pre-conversion needs to be equal to the sum of the data in post-conversion and the fallout.
A difference should be retraceable.
At the conversion of software en JCL no functionality is allowed to be lost.
2. Controllability
Mostly plays a role in the conversion process itself and needs to be taken along

Specific points of attention for migration

explicitly as a design step, in both the source code as the data conversion. For data conversion the principle of hash totals and additional record counts can be used amongst other things.

3. Performance

In case of a data migration the duration of the conversion could unexpectedly disturb the migration process. Mostly a weekend is reserved for this. The data conversion should fit in to this however.

4. Reliability

Has a fallback scenario been considered if the migration needs to be reversed during the execution?

Is it possible to pause and restart the data migration?

4 TEST APPROACH OF MIGRATIONS

4.1 Why test

Some organisations are convinced on forehand of the use of testing migrations. But other organisations see the migration as a technical process that, especially if preformed in an automated way, will lead to faultless applications. For the latter it therefore happens in practice that, just to be certain, some little tests are performed near the end. If the testers find odd things, and they often do!, it may happen that a test phase is added after all. This way the implementation date overruns its time, sometimes by months, and extra budget is needed for testers and developers and users.

Experiences from practice of all sorts have been documented in chapter 5 'Experiences testing during migrations'.

The reason for testing is that the people responsible for the business process that is supported by IT, want to be certain that the business process continues undisturbed after the migration. Practice shows that testing often discovers important defects that would otherwise have occurred in the business process and outside of the guarantee period.

Another reason for testing is that a formal approval needs to be given for the executed conversion/migration activities.

In this chapter the TMap-description manner [1] [3] is followed. The indications described in TMap [1] and [3] will not be repeated. Only the migration specific indications are described for:

- Test assignment/assignment formulation including test project risks
- Test basis
- Test strategy

and the standard TMap-cornerstones:

- Life-cycle
- Techniques
- Organisation
- Infrastructure

4.2 Test assignment / Assignment formulation

In the test assignment the expectations of customer and supplier are made explicit. Per TMap-subject in the assignment formulation (including test project risks) the following migration specific subjects can be appointed:

Area of consideration/scope

- Is the testing of the interfaces with other systems part of the assignment? If so, which ones?
- Will the system documentation be migrated as well and therefore 'tested'?
- Will the testware (amongst other things manual and/or automated scripts) be migrated and therefore 'tested'?
- Will the service management tools be migrated and tested?

Goal

- Should it be tested whether the software works conform the pre-migration or conform the documentation/specification

Test approach of migrations

- For the detailing of the goal of the test assignment the acceptance criteria of the migration play an important role. Important for an acceptance criteria is:
- The requirement, for example meet the security requirement
- The way of demonstrating, let a hacker do his/her work for a week.
- The allowed deviation of the requirement, for example only findings with a lower category of severity.

Often found acceptance criteria:

- functional (= correctness and completeness) the same as for the migration with as variants for the demonstration:
 - simulate a day or week of production
 - let all functions perform at least one transaction
 - let all incoming and outgoing interfaces handle every kind of message
 - the available test cases have been executed completely (and the testware is adjusted)
- performance at least equal (batch and on-line)
- for a data conversion the control totals need to be correct with exception of explainable differences, whereby these differences need to be approved by an institute (for example Internal Accountancy Service)
- in some cases the correctness of the data conversion needs to be proven at a very detailed level. To prove this, an extra tool is often required.
- the look-and-feel of the screens needs to be accepted by the users, based on the substantiated opinion of the user without establishing more precise criteria beforehand
- maintenance programmers need to be able to read/maintain the converted software, based on the substantiated opinion of the maintenance programmers without establishing more precise criteria beforehand
- all automated processes need to be start able in planned order and on planned moments and work together properly
- the security needs to remain compliant with the given standards

The allowed degree of deviation also depends of the contractual situation. If there is a limited period of guarantee it could also pay to let the subcontractor solve all unimportant findings. This might be at the expense of the lead-time.

Preconditions/starting points

- Can functional changes be processed throughout the migration period or does a 'frozen period' exist?
- Separate test environment for each test level.
- The closer the test level is to production the more the test environment needs to be like the production environment. If hardware migration is also an issue this could be a bottleneck. A measure is to use the target machine for production first in the acceptance test.
- The test sets that are found in the testware assessment do not contain inaccuracies and operate on the pre-migration production environment.
- Existing errors in the pre-version will not be fixed during the migration.
- Domain experts are available to judge the testware and findings and to fill in for the missing test cases; this will be done within a fixed term.
- The converted production data are available for the post-migration tests.
- Back-up and restore of the data are possible and guaranteed in both the old and the new environment
- For performance measurements the correct tools/licences need to be present.
- Test databases are made available from an agreed moment on.
- Self-built tools for for example file extraction are made available after being tested.

Test approach of migrations

- Order of delivery of program conversion and data conversion optimal for test planning: in general testing is faster on the already converted production data; no extra activity is needed then for filling the test database.

The test project risks at least consist of the chance that the assumptions and precondition are not sufficiently defined on time. For every possible risk the damage/consequences and the measure need to be clear.

Specifically to be mentioned:

Project risk	Measure
Too little users available for testing.	The available time of the user needs to be made maximally effective by letting them do as little findings as possible. This can be achieved by having professional testers do the initial testing. They will establish many obvious findings.
Users create insufficient correct and in-depth test cases/scripts	Make sufficient agreements in the contract and measure these during the execution. Then recall and if necessary escalate during the migration project.
The test approach demands that the test actions on the pre- and post environment will be executed in exactly the same manner and in the same order. The chance on execution errors is large here.	Set up detailed test scripts and a detailed test scenario. Shorten the time between the execution of the same action (preferably parallel) Let two testers (users) execute the test scripts, also to make it possible to do checks amongst each other.

The standard project risks mentioned in TMap (see [1], paragraph 17.5) also need to be considered.

4.3 Test basis

In a migration test assignment it is of important to agree on what the test basis is. In most situations the migration assignment is to migrate 1-on-1. In that situation the behaviour of the system in the pre-migration situation is in fact the test basis. This behaviour can be determined in a day or week of production simulation (based on possible acceptance criteria) but also by using the available test scripts. If existing testware is missing or the depth is insufficient, a functional design (FD) could function as the source for creating the situation to be tested. The output in the pre situation is the prediction for the post situation and not the prediction conform the functional design, unless agreed differently. If a FD is missing, the knowledge of the users need to be used.

4.4 Test strategy

To set up a solid and accepted test strategy the next steps are necessary:

1. Set up and classify product risk analysis
2. Assign risks to quality attributes and classify the risks of quality attributes
3. Assign risks and quality attributes to test levels with indication test depth

For the step Set up product risk analysis there are migration specific indications, see below. Regarding the use of test levels practical indications are presented.

4.4.1 Migration specific clues for establishing product risk analysis

The risks are determined with all people involved; see the technique risk analysis in TMap test Topics [3] for more information. Risk is the product of damage and the chance for damage (chance of failure). The chance of damage is after all the reason for testing. For the damage side no specifics can be mentioned for migrations. For the migration specific chances of failure the non-technical issues are presented below. For the more technical focused chances of failure we refer to appendix 7 Checklist technical risk.

Generic risk points of failure:

- In general for migrations it can be said that the size of the technology jump is proportional to the size of the chance of failure because of unfamiliarity with the new technology.
- Manual converting is proportional to the chance of faults and the diversity in faults.
- The manual converter needs to have knowledge of the target language. Otherwise many faults will arise. The same goes for setting up the rules of conversion for the automated conversion. The rules are applied to the conversion tool after all.
- If data in the pre-migration database originates from different companies/departments (historic data migrations) the integrity rules are often weakened and workarounds are installed in the program where necessary. In that case it cannot be guaranteed that the migrated combination works (functional/performance). This mostly expresses itself in the batch functions using all data.
- A mixture of language in one transaction (e.g. a Cobol-routine calls upon an assembler-routine) has potential problems that are known on forehand.
- Unexpected inaccuracies in the data can lead to incorrect processing. For example: in the pre-conversion database the symbols for gender are M(ale)/F(emale)/U(nknown, but U is not known in the documentation. At the conversion of the data the U may be forgotten.
- After approving the program conversion and the data conversion the combination can still deliver errors. A cause is often lack of data quality combined with work-arounds in the pre-conversion software.
- If the release is not read/understood or the impact analysis de facto is not possible, there is a large chance that an error causing 'feature' mentioned in the release notes is overlooked.
- After conversion of data to another database structure the size could have increased. This needs to be processed in buffer sizing, etc.
- If a database management system is migrated and the data structures in the pre- and post situation are different than extra attention is needed for data mapping. If the available data mapping knowledge is insufficient many surprises will occur.
- Upgrades of modern tools e.g. power builder can lead to incorrect processing.
- Programs could contain more rules after the conversion by a conversion tool.
- If more complex possibilities of the programming statements are used in the pre-situation it is probable that these will not be translated properly to the post-migration situation. The conversion tool does not have to be organised correctly yet on that more complex possibility.
- If retrospective mutations are possible this could cause problems for mutations that go further back than the conversion date.
- Is it still possible after the migration to perform a retrospective mutation correctly?

Test approach of migrations

- Conversion of diacritic (for example Å, ü) does not work correctly.
- JCL is not converted correctly: files cannot be opened.
- It turns out that error handling procedures cannot be called upon

For the translation to the risky quality attributes the list of quality attributes that can possibly be affected in chapter 3.1 can be used.

4.4.2 *Test strategy considerations*

Determining the test strategy is determining the test effort based on the calculated risks, given the available budget and lead-time.

Below practical considerations are shown:

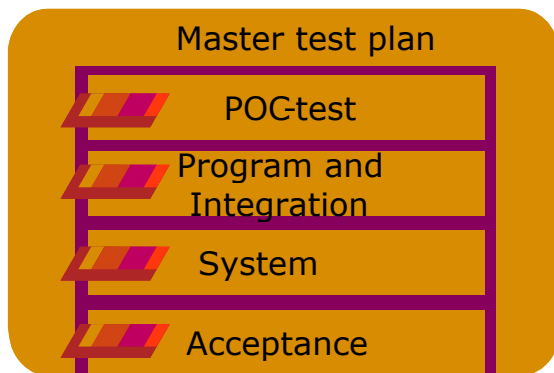
- The migration strategy is part of starting point. After all setting a proper foundation requires more test effort than enforcing a temporary solution, for example code reviews are not necessary for a temporary solution.
- Determining the most risky system parts needs to involve the entire system, so also the annual batch functions like the prolongation run for insurances and the year balancing of financial systems. The error chance because of potential data pollution is usually not small. Especially for the prolongation run damage can easily increase because of an error.
- If part of the migration is outsourced the repairs during the warranty period are charged to the converting party. In that case it pays off to test more to do findings of lesser severity.

The test objective regarding spreading responsibilities could be relevant for determining the strategy:

- Test for discharge of the converter of software, data, etc.
- Test for discharge of the system integrator
- Test for the usefulness in the organisation (suitability/usability)

A possible approach for the division of the test effort over different systems in a migration: look at the technical complexity, give the complex ones more test attention than the less complex ones by explicitly testing the complex ones with designed test cases and the less complex ones with a limited number of user scenarios.

4.4.3 Master test plan en test level



A specific extra test level common for migration projects is the POC-test. A master test plan covers the possible test levels:

POC-test

In the Proof-of-concept test it is tried whether the proposed migration approach will work or not. The POC-test contains the conversion of all sorts of software in the information system. A limited number of each component sort is brought along. It is sometimes stated: by converting 10% of the system we find 90% of the problems.

It is recommended to involve the converted production database in this test. For many findings the converter needs to improve the conversion process and the POC-test needs to be repeated.

If performance is an issue, the performance can be tested as well. Possibly by measuring the aspect of efficiency.

Program test en program integration test

Is the conversion complete and correct and do the programs and data base structure work together in a stable way again? This test is often carried out per conversion cluster. There also is a possibility for program code review. A cluster can be the test of data conversion software.

On the level of the converted cluster and test of the conversion software. In this test level the code review of each program can be carried out.

System test en system integration test

Test of the functional correctness of the integrated software and database. This test level can include performance tests and/or efficiency measurements. The test is preferably performed in a production like environment instead of a laboratory environment. The test cases contain the already existing testware or will be designed specifically by reproducing (a selection of) production situations. The test can contain the executing of yearly important batch functions (for example prolongation run and year transfer). This needs a time travel test.

If possible as many internal and external interfaces as possible are tested as well. For this the chain test is used.

Test approach of migrations

Acceptance test

The trial conversion of the production data can be part of the acceptance test. Testing has to be done with the converted production data the functional correctness and completeness with regard to the production situations. If it is not yet executed the chain test with other systems should be included here.

This is also the possibility for system managers to test things like (JCL)-production scripts, parameter settings, security issues, typical system manager tools (sniffers, data checks) and job scheduling (PAT).

Performance test (batch and on-line) can be included in this test as well.

4.4.4 Detailed test plan

The test strategy on the level of detail test plan will be executed with the standard TMap approach and lead to test units with a predefined test approach and effort in hours.

The next steps apply to the POC:

1. Take representative parts that form a testable entity from all different sorts of software (screen handling, database transaction, processing logic (on-line/batch), JCL, etc.) (Representative = all used language constructions, preferably the most complex programs, programs from different development teams.)
2. Convert those with the chosen approach,
3. Perform a review of the code to validate the conversion rules
4. Integrate the parts to a testable entity
If required execute the trial conversion of the production data
5. Testing of the functional operations and possibly measuring the efficiency of the application
6. If the conversion is also relatively new for the converter, it is advisable to test the outlines shallow at first (e.g. with the DCT) and go deeper with a larger team if it works successfully.

An additional way of prioritising the programs to be tested in the program test is the following:

For the conversion of program source code the following degree of increasing chance of failure can be distinguished:

- 1 one-on-one (automated)
- 2 one-on-one tricked (automated)
- 3 one-on-one tricked (manual)
- 4 complex (manual)

By counting per program how often one of the four situations occurs the degree of chance of failure can be determined per program. A mutual order of chance of failure will come out anyway.

4.5 Test life-cycle

Life-cycle MTP

The life-cycle for MTP conform TMap is an excellent starting point.

"Inventory and assessment available testware" needs to be taken in as extra activity. This activity can often be combined with the Orientation activity from the MTP.

Life-cycle DTPs

The life-cycle is according to TMap. In the specification phase test scripts are prepared without a result prediction. The execution prediction is after all the execution of the pre migration system. The execution of test cases in the post migration systems preferably happens simultaneously in the pre migration system.

4.6 Techniques and test types

Below the techniques for which something specific should be mentioned for testing migrations are listed. For those who do not know TMap a detailed source statement is given.

4.6.1 Budget aspects

It is difficult to get consistent practical numbers. The situations in which conclusions can be made are mentioned below.

The sec observation are mentioned in chapter 5 Experiences testing during migrations

Time needed for composing (master) test plan

Composing a master test plan in a organisation that yet needs to be convinced of testing takes significantly more time than in an organisation that already understands the value of testing. One of the reasons is that the risk analysis for "desired" testing is created in less time because less consultation is needed. Low: 60 hours - High: 160 hours.

Time needed for testing

Guidelines are not easy to give. Hopefully the future will provide more experience. Below the first conclusions for functional testing.

1 Top-down

A first re-calculation indicates that the black box testing effort (system (integration) test) is equally as big as the program conversion effort (including limited testing). An acceptance test with participation of many users costs equally as much as the system test. In this situation the total black box testing costs twice as much as the conversion.

2 Bottom-up

Experience in one project for the functional test:

- composing test scripts
- executing null measurements (manual)
- executing end measurements (manual)

costs 1,6 hours/function point (excluding overhead).

3 Upper limits

The upper limit of the functional test effort can be determined by using the budget approach for functional testing. The design and the execution of the test set costs approximately 2 hours/function point (including overhead). The number of function points could be counted based on the data model (see TMap Test Topics [3]).

4.6.2 Review technique

Review of the program code by the future maintenance developers.

4.6.3 *Test specification and execution techniques*

The test techniques mentioned here are described in detail in TMap [1] & [3]. Conversion specific cases are discussed below.

It can be considered to use and convert the regression test files from the pre situation.

Checklist

An addition to a checklist:

Are all changes mentioned in the release notes checked on validity for the application and if so are measures taken? Sometimes the technical specialist did but the application expert did not!

Are further situation specifically based on the typical chances of error.

Exploratory testing

If there are no test scripts available and the existing processing also cannot be input exploratory testing is a possibility.

Exploratory testing can be used well for the POC-test. Within this the needed skill knowledge of the existing system is preferably completed with knowledge of the conversion approach. (Manual/Automated). For an extensive description of ET see TMap Test Topics [3], chapter 14.

Photo comparison technique

The photo comparison technique is based on the comparing of the behaviour of the system in pre- and post migration situation.

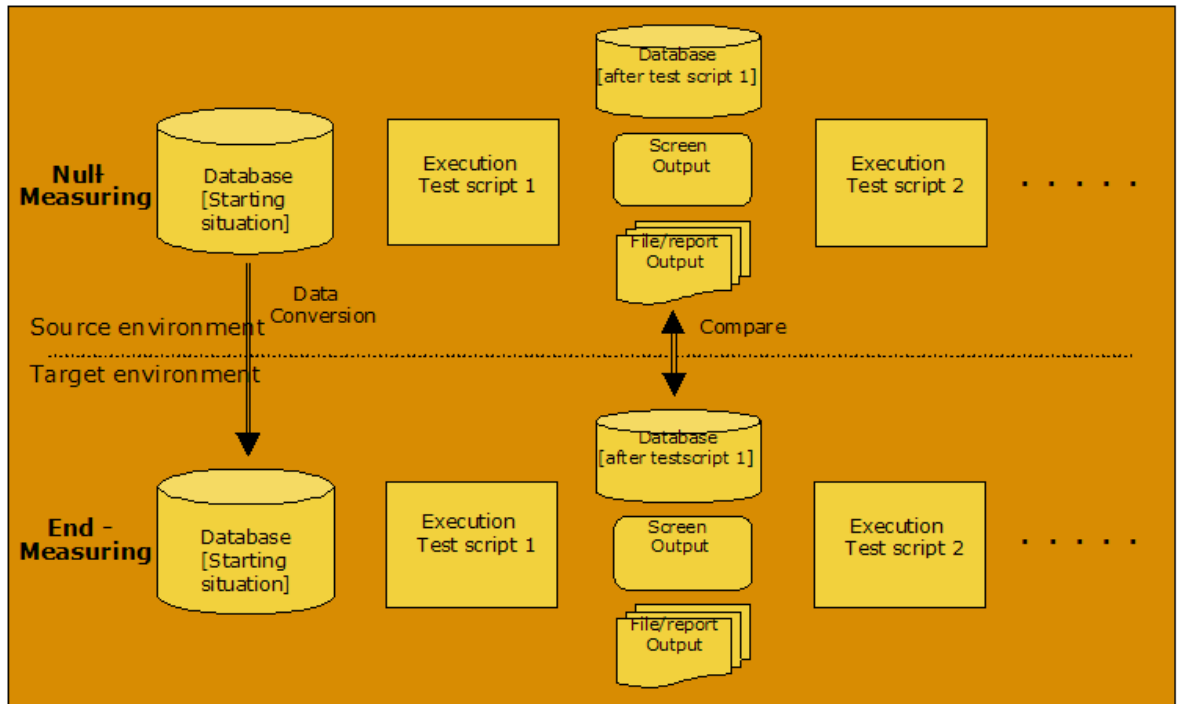


Figure: Automated photo comparison

The behaviour of the pre-migration situation can be recorded automated beforehand (null measurement) and played in the post migration situation (end measurement). With automated executing lead time can be won in the project and the test sets could be used for maintenance testing.

In a manual variation two users, one on the pre-, the other on the post situation, can perform similar transactions, compare the possible output and check with the read functionality if the database is brought up to date identically.

Real-life test

In the real-life test the actual use of the system is simulated (=100% real-life). The choice can be made to test with a depicted image of the reality, for example with 20% of the number of business activities.

The real-life test can be applied for several purposes:

- 1 For performance testing the simulation needs to be 100%.
- 2 For stress testing even more than 100%.
- 3 For functional testing the range is between 1 en 100%, dependant on the risk.

The real-life use of the system can be determined by logging user transactions in the pre-migration situation during a representative period.

Static test of conversion script

Developers, testers, system managers and users judge the conversion script.

Time travelling

For time travelling an adjustable processing date needs to be used for travelling through time. This could be useful for testing the year closing for example. The system should have the possibility to set a processing date. If necessary this facility could be built in in the conversion.

4.6.4 Test forms

Chain test

Assuming that a migration does not involve organisational changes, it is sufficient in the chain test to include all sorts of interface records. This is often only possible in the production environment, but it is recommended to execute this in the system test environment as often as possible. See for general instruction for chain test TMap Test Topics [3], chapter 15.

4.7 Organisational aspects

4.7.1 Organisations involved

The departments/organisations that are always involved in IT-projects are also involved in a migration. For a data conversion a department like Internal Accountants Service or Controlling need to be involved.

4.7.2 Test- en application expertise

Users know how the system behaves before the migration. After the migration the system should do the same. The users do however need to do their daily work. There is usually little capacity left for testing. It should be recommended to use this capacity for indicating the broad outlines of the testing. Additional testers can work out the details by executing the test cases on the existing system. The execution prediction for the situation after the migration is than also known.

4.7.3 Defect procedure

Arrange for the defects to go to the converter quickly. A defect in the conversion rules and/or software needs to be adjusted in every place it occurs after all, also in the software already delivered. The defect management must be accessible for all parties, and each party needs to be able to execute the actions that go with the role.

4.8 Test infrastructure aspects

4.8.1 Test environment files

Separated test environments are necessary for the developer/converter (if needed with stubs and drivers), the system test and for the acceptance test (production size).

For the manual comparison the test environments for the pre- and post situation are needed at the same time. For automated photo comparisons test environments for pre- and post situations are convenient to check indistinctness in the post system in the pre migration system right away.

For performance tests of batches production volume is needed.

Sometimes chain testing with other applications is only possible in the acceptance environment.

4.8.2 Tools

Below the tool sorts that are used in practice are presented.

File comparison

The most ideal tool is capable (using aid programs) to compare the content of the databases in the pre- and post situation and clearly indicate possible differences. Databases often need to be converted to flat files. Flat files can be compared. Well-known compare tools are WinDiff and Sdiff (Unix)

Defect Management

Standard: has to be able to deal with thousands of defects and be used from multiple locations, especially for outsourced conversion work. View the knowledge database for known tools at the TMap® ToolCenter.

Database reading software

This type of software usually is self-built. Sometimes a link from the comparing program is possible to the databases in the pre- and post situations through ODBC's (ODBC= Open Database Connection). Concrete example: a MSAccess-program that links to pre- and post situation database, both through ODBC. Often there will be additional specific development in order to get from a database structure to a flat (sequential) file.

Intercept users transactions in production

Tools exist for the interception of user transactions in production.

Record & playback

Bring record and playback tools into action may be useful for a photo comparison test. The look-and-feel of the screens should correspond or be easily translated to each other. The data driven approach TAKT [6] may be of use. The SC TMap® Tool Centre has extensive expertise on this.

Stubs and drivers

Test approach of migrations

Can be necessary for the programmers' tests and in the system test.

Test coverage

If undesired uncertainty exists on the depth of testing of the developed test set it can be considered to measure the test coverage. Providers can be found via the Internet.

MigrationWare has developed a tool as part of the conversion software, see 6.1 Program code for additional info.

4.8.3 *Office environment*

Access for tester to pre- and post situation to execute test scripts, preferably at the same time and at the same location. Defects in the post situation can then be tested in the pre situation.

5 EXPERIENCES TESTING DURING MIGRATIONS

During the research many practical experiences have been gathered. These are mentioned in paragraph 5.1. The experience that might be useful for budgeting is presented in paragraph 5.2.

5.1 What do testers find in migration practice?

In an atmosphere of "just" a technical conversion with preservation of functionality sometimes schedules for testing are made that turn out to be factors too short. In other cases migration is prepared optimally. Overrunning of time and budget are then very limited. Below some experiences:

- We started according to plan with one tester for three months and ended via two testers after six months with three testers after nine months.
- The testing of the JCL-scripts was grossly underestimated: instead of 300 hours, 1000 hours were spent testing the JCL-adjustments.
- Installing the system in the test environment so that functionally testing could be done did not take the estimated two weeks, but 10 weeks for the installation and an additional 12 weeks for solving the technical problems. So after 22 instead of the planned two weeks the functional tests could be started.
- In project plan: Lead time of migration 6 weeks without testing. Testing was not necessary according to the converter because it was just a technical migration from PC/OS to another type. After one day of rough testing by a tester problems appeared, after 5 man-days by specialists a hundred serious defects were found. In the end the migration took three months, with two teams and 12 people per team. A team consisted of at least a technical converter, an application expert and a tester sometimes supplemented by a user.
- For a version upgrade of a transaction monitor technical problems occurred during testing in the functioning of the converted software. The release notes of the version upgrade had not been read through carefully. In the plan repairing was not considered. Besides the period for repairs there were three weeks in advance of political debate on the execution of the repairs.
- Many defects in System test and Acceptance test. (no interpretation data available)
- De conversion software is different per type of component. At a conversion of approximately 2700 programs approximately 3000 small defects were found, especially in the batch (lists). Small means easily solved, approximately 2,5 hours per defect.
- During a very well prepared program migration and an migration (with data cleaned up in advance) very few defects were found in the photo comparison technique.
- During a Cobol version migration (small step) few defects were found.

Experiences testing during migrations

5.2 Experiences with budget and realisation

Several project experiences in a row.

5.2.1 Realisation formulating master test plan

Situation 1: 160 hours. Contains amongst other things strategy and limited risk analysis. Division: 50% into the organisation, 50% writing it down.

Situation 2: 60 hours, with users that are experienced in risk analysis and an experienced test manager.

5.2.2 Plan and/or realisation migration project including testing

Below several project experiences are summed up. Used concepts:

- POC = Proof Of Concept, the most critical aspects of the program and data conversion are executed
- Basic conversion = the conversion of the source code to the target system without or with limited testing. This conversion is often done by a sub contractor.
- System integration (SI)= bringing together all system components, for example: software, converted production data, the new hardware and system software, and the management tools. This phase could contain testing and repairs.

Project1

Size: approx 20.000 hours

Source: Project administration

POC	6%
Basic conversion	25%
Testing (FAT PAT)	41%
System integration	28%
Total	100%

Project2

Size approx 10.000 hours

Source: Plan

No POC in the numbers

Basic conversion:	25%
Testing (FAT PAT)	31%
System integration	44%
Total	100%

Experiences testing during migrations

Project3

Size approx 10.000 hours

Source: offer

POC (hours not clear)

Basic conversion

Automated	38%
Manual	17%
Testing	38%
System integration	9%

Project 4

Size approx 7500 hours

Source: Project administration

POC	18%
Basic conversion	17%
Testing FAT PAT	5%
System integration test	35%
Bug-fixing in SI	25%

Project 5

Size: unknown

Source: Test project manager

The system (integration) test cost as many hours as the manual conversion (including the program test and a limited integration test). Besides this a user acceptance test of the same size (in hours) as the system test has been performed.

That seems to be a lot, but is caused by the fact that many users and application specialists need to be consulted.

Project 6

Size: approx 3000 hours

Source: Test manager

Conversion: Testing= 1 : 2

Experience in the Millennium/Year2000 migration projects

Proportion migration: testing= 50:50.

6 APPENDIX: THE WORLD OF MIGRATION

6.1 Program code

The following ways of conversion/migration of program code are known:

Migrationware (www.migrationware.com)

Migrationware uses own built conversion tools and therefore uses cheaper work force (South-Africa).

Migrationware has, besides conversion tools, also got tools for the photo comparison technique, also for on-line functions. To avoid comparison of the databases the call to the database is trailed. The supplier claims that the comparison of the pre and post migration situation is simpler.

In this tool the coverage of the test set is also measured. 70% coverage of the paths is high coverage! This works as follows:

For each path in the software a dedicated precompiler creates a place in an external array and a piece of code for each path. If during the execution of the test set the path is touched this will be indicated in the array. This array is external compared to the program, by which the effect of more test sets can be accumulated. It can then be viewed on-line what the effect of test sets is on the coverage.

Cornerstone (www.cornerstone.nl)

Cornerstone uses their G4 technology that can be converted from any arbitrary language to another arbitrary language:

The basis for our advanced G4/Technology consists of a specific **dynamic parser** that is generated for analysing the sources of the client. The resulting meta-information is stored in a database allowing for all kind of queries and reports, in order to meet the individual user information needs.

The G4/Technology is programming language independent. No matter what programming language, the **Cornerstone Competence Centre (CCC)** can help out with any problem the client may encounter regarding automatic documentation, software analysis, [modernization](#), [renovation](#), [software maintenance](#), [conversion](#), [migration](#), [automated \(mass-\)changes](#), [beautification](#) that have to be performed. It helps to document/visualize all interactions within the entire application system on a higher meta level. G4/Technology is effective at a technical, control and managerial level.

After the parser is generated, we process the application system source code (e.g. JCL, Cobol, screen-maps, file definitions, DDL). The resulting information called 'meta-data' is stored in a database addressed as the G4/Repository. No information is lost. This enables us to answer any question the customer might have about their application systems. This technology also enables clients to intervene in the source code to perform changes or conversions. The changes may vary from semantically to syntactical, from beautification to language conversion. The reason for change may be migration to another platform or a total modernization of the entire application system.

6.2 Data(base) management system

Tresco (www.tresco.nl)

Tresco has an approach and solution that migrates a data(base) management system without the need to adjust the source code.

Basic principle: The pre-compiler to the source system data(base) management system is replaced by a smart pre-compiler that takes care of the translation of data structures and the link to the target data(base) management system.

6.3 Data conversion/data quality

- Capgemini has its own software for measuring data quality
<http://www.testnet.org/Produktie/Bibliotheek/Thema20050323/Datakwaliteit%20en%20Testen%20NK.pdf>
- There are measuring models: MIT/ TDQM
<http://web.mit.edu/tdqm/www/index.shtml> and Maintain/KING
<http://www.testnet.org/Produktie/Bibliotheek/Thema20050323/Testen%20van%20datakwaliteit.pdf>
- There are specialised companies: www.Arrix.nl, www.HumanInference.nl in Arnhem, The Netherlands.

7 CHECKLIST TECHNICAL RISKS

This checklist is based on experiences in practice as they were found while executing the research for this white paper and in the document Experiences with migration projects [4].

In the white paper Experiences with migration projects [4] besides the generic points of attention for the testing many technical issues are mentioned. In version 2.0 (19 October 2004) technical issues are mentioned for:

1. DB2 to Sybase
2. IDEAL to ANSI Cobol Wintel
3. S390 Cobol/2 to AS/400 Cobol
4. CSP to Cobol-85

These issues could each signify a chance of failure.

During the inventory for this white paper the additions below have been found:

From	To	Issues
KEY	Coolgen	few issues
IDMS	DB2	<ul style="list-style-type: none"> • null-values: allowed in IDMS in DB2 mostly not • non numeric values allowed in IDMS and not in DB2 • date-issue: IDMS has no date format because of which misuse of the fields is possible • sub files in IDMS, not simply convertible to DB2 • cleaning: clean logic missing records in IDMS before moving to DB2 • development standards: the used IDMS-coding standards are not 1:1 transferable to DB2. • Order of screen presentation possibly no longer correct (because IDMS used chains and DB2 does not) • Commit (batch and on-line): commit in DB2 commit whole transaction, in IDMS just the run-unit. Creates problem when a transaction has more than one run-unit open at the same time DB2 needs more CPU-cycles than IDMS, which creates: <ul style="list-style-type: none"> • chance of performance loss • a higher calculation centre bill • Can deteriorate performance other systems
OS/2	Unix	is not 1:1 by the unexpected large amount of Unix-possibilities amongst which case-sensitivity
IBM MVS	non-MVS	is from EBCDIC to ASCII, consequences for sorting in files and sometimes consequences for the operating of programs that use typical EBCDIC characteristics
Cobol	Something else	packed stored numeric values redefines with much smartness, locking-principles (one on one copying of the locking can lead to wait times and therefore performance problems)

Literature

8 LITERATURE

- [1] Testen volgens TMap®, 1999, ISBN 90 72194 58 6, publisher Tutein Nolthenius (UTN)
- [2] Mikado: Migreren zonder heibel in de business, 2005, Martin van Capelleveen, internal Sogeti NL
- [3] TMap® Test Topics, 2004, ISBN 9072194 70 5, publisher Tutein Nolthenius (UTN)
- [4] Ervaringen met Migratie-projecten, 2005, Norbert Mimpfen, internal Sogeti NL
- [5] Acceptatietest van gegevensconversie, 2001, Arlan Lesterhuis, internal Sogeti NL
- [6] Automatisering van de testuitvoering, 2001, B.Broekman, C. Hoos, M.Paap, SDU/Tenhagestam ISBN 9044001035