Guidelines for legacy repository migration

Miguel Ferreira; KEEP SOLUTIONS; Braga, Portugal
José Carlos Ramalho; Department of Informatics, University of Minho; Braga, Portugal
Luís Faria; KEEP SOLUTIONS; Braga, Portugal

Abstract

Several institutions are currently running long-term digital repositories that have been in operation for several years now. Some of these systems are approaching the end of their life spans and will soon be replaced by the next-generation of long-term digital repository systems. This will unavoidably imply the migration of millions of files, metadata records and terabytes of data from the legacy repository to the newly adopted one. Because of the large scale of this operation, this procedure needs careful planning, validation and support.

This paper provides guidelines to support the migration from legacy repository systems by describing the stages, activities and associated risks that comprise this type of endeavor. The presented guidelines are based on a combination of 13 existing methodologies that have been surveyed and unified into a comprehensive multistep methodology.

Introduction

There are many reasons by which organizations decide to migrate to new a repository system, for example:

- Repository system does not cope well with current business needs (e.g. lacks desired characteristics like performance, capacity, interoperability, usability or others);
- Budget cuts mandate that a new, more financially sustainable repository is adopted;
- The repository vendor or supporter ceased to exist (i.e. the repository is no longer supported);
- Repository vendor or supporter does not provide a satisfactory level of support services;
- The technological environment needed by the repository system is no longer supported (e.g. security updates are no longer available for the supporting operating system).

Several scenarios can be considered examples of legacy repository migration projects, for example [1]:

- Migration from a relatively simple system into another system;
- Upgrading a system to a new version of the same system;
- Converge multiple systems into a single composite system;
- Critical system migration that requires the migration to be rolled out over a period of time without interruption of operations;
- Multiple concurrent systems migrations and consolidation efforts (this is often referred to as "IT Transformation").

It might appear that any two systems that maintain the same sort of information must be doing very similar things underneath the hood and, therefore, should map from one to another with ease. However, this is hardly ever the case. Legacy systems have historically proven to be far too lenient with respect to enforcing integrity at the data level [6]. Fields that typically should be populated from a controlled list of values tend not to be validated by the system, and therefore the database ends up with unexpected values that require exceptional handling during migration. Another common problem has to do with the theoretical design differences between hierarchical and relational systems. Two of the cornerstones of hierarchical systems, namely de-normalization and redundant storage are strategies that make the relational purist cringe. [6]

Additional difficulties may be encountered while migrating information from one system to the other, for example:

- Extracting information from the legacy system can be extremely complex, especially in the case where the functionality to export information does not exist, technical support is unable to provide the necessary help, documentation is scarce or incomplete, the organization does not have the necessary system level credentials to gain actual access to the data;
- Mapping between the previous semantic structures to the ones of the new repository might be difficult to attain, or impossible when these structures are highly incompatible (i.e. data loss will take place);
- The process of transforming and/or cleansing data during the migration process is prone to errors caused by incorrect settings or bad programming;
- The necessary validation procedures can be extremely difficult to design or automate.

Although migrating data can be a fairly time-consuming and risky process, the benefits can be worth the cost, as legacy systems do not need to be maintained any longer. Although migrating from legacy systems is a major research and business issue, there are few comprehensive approaches to migration. Given the bewildering array of legacy information systems in operation and the problems they pose, it seems unlikely that a single generic migration method would be suitable for all systems. However, a set of comprehensive guidelines to drive migration is essential [2].

Related Work

In the context of Information Technologies, the term migration may mean a lot of different things. Considering the context of digital preservation alone, concepts such as file migration, media migration, format migration, repository migration, data or metadata migration/conversion are commonly found in the specialized literature. However, all of these terms mean very different things and may comprise very distinct approaches in the way they are conducted.

In the context of digital preservation, the term “repository” is often used with different meanings, but in the context of this paper, the definition of repository is a system composed of software and hardware that is set up to follow certain rules or policies and that is responsible for safekeeping and managing digital information.

A repository usually entails several types of digital information, for instance:

- **Data** – usually the most important asset managed by the repository, i.e. this is the actual information that users are interested in and expect to be kept safe and accessible (e.g. images, audio, video, documents, datasets, 3D objects, etc.);
- **Metadata** – information about data managed by the repository. Metadata fulfills many goals, e.g. supports data discovery, ensures authenticity and provenance, provides characterization, etc.;
- **System specific information** – information highly dependent on the information system, that is often automatically generated and it is intrinsically necessary for the system to function (e.g. configurations, logs, indexes, user information, branding files, etc.).

"Repository migration" is the process of transferring digital information between two or more information systems, whether this is data, metadata or any other kind of information considered to be relevant to the continuity of the organization or individual in charge of that information.

Two classes for repository migration approaches can be considered. The first class is "component migration" in which the legacy systems are broken down into independent components and each component is migrated separately. There will be a period of transition where both legacy and the new platform have to be online and to work together. Two strategies will arise, "phased interoperability" and "parallel operations". Both of these need the data to be shared via "database gateways", replicated on the two platforms, or sliced into separate independent domains to be migrated gradually to a new platform [15].

The second class of migration approaches is the "system migration" approach in which the whole legacy system and the data are transferred to the new platform in a single step. There are two subclasses to this approach: "no value added" (in which the system remains practically the same, either by emulation or simple upgrade) and "value added" (where the system acquires a considerable number of new capabilities) [15].

This last approach, value added migration, leads to changes in the user interface, the database and the program logic. Although, migration may be more complex in this situation, its long-term benefits will be much greater. It may offer more flexibility, better system understanding, easier maintenance, and reduced costs [15].

Approach

In order to develop these guidelines, we adopted the following four-step procedure:

Step 1 - Survey existing best practices documentation

This step consisted of collecting existing best practice materials from recent years. The survey revealed a significantly mature research field in which the phenomenon of repository migration is already well framed, i.e. the Legacy Information System Evolution [2, 3, 6, 9–11]. Additionally, several whitepapers, communications and technical reports from the IT industry were also surveyed and taken into consideration as they provide valuable hands-on information necessary to support these guidelines [1, 4, 5, 7, 11–14, 16].

Step 2 - Identification of repository migration methodologies

Several of the research and technical documents collected in the previous step depicted methodologies on how to perform legacy information systems migration. Some of these methodologies were very simple and generic, while others were very detailed and domain specific. This step consisted of the analysis and evaluation of all the previously collected documentation and the selection of the ones that included well-grounded information on how to perform repository migrations.

Step 3 - Comparison of repository migration methodologies

Some of the methodologies identified in the previous step were very simplistic, being composed by merely four generic steps that could very well be applied to any software development process, while others were extremely detailed in the tasks they depicted (down to the product name and version number). In order to systematically compare all of these approaches we have created a comparison matrix where all the common steps described in each of the methodologies were aligned for easier comparison.

Step 4 - Creation of a unified repository migration methodology

The next step in the creation of these guidelines was the classification and generalization of all the activities found on the surveyed methodologies. This process allowed us to combine all the approaches into a single unified methodology that comprised the steps included in all the other surveyed methodologies. We also made small adjustments to the terminology to make it more compatible with the jargon used in the digital preservation domain.

Repository migration best practice guidelines

These guidelines are, above all, a formalization exercise that aims to identify and describe the most important steps in a repository migration process. They do not intend to be prescriptive or even complete but instead they aspire to provide enough information to any vendor, customer or IT specialist to crosscheck that the most important steps have been addressed during the planning and execution stages of a repository migration project.

Table 1 summarizes the unified methodology for a successful legacy repository migration. The methodology is composed by 7 stages, each of these composed by a series of activities.

The following sections describe each of the stages and activities included in these guidelines.

Analysis & consultation

The first step in a repository migration process is to gain insight on the systems, data characteristics and the needs of all interested parties in order to define the most appropriate migration strategy and formalize all the necessary requirements. This includes a deep analysis of both legacy and target systems, the data to be migrated, and the expected business needs to be met. The quality of the analysis stage will dictate the success or failure of the overall migration project.

1A - Characterization of legacy environment

The first step in preparing a repository migration should be the assessment of the legacy system(s) technology environment. Repository migration requires a complete understanding of all the involved technologies, including hardware, networks, software, programming languages, data structures, services, servers and time requirements (e.g. availability of the system).

To be able to size and plan the migration process, as well as setting accurate budgets and timelines, one must understand the complexity, relationships, quality, and volume of the legacy system and its data. This will enable the definition of appropriate

requirements such as replication needs, project schedule, system response times, vendors that need to be contacted, and the hardware configuration necessary to host legacy and upcoming data.
The management costs of the legacy technological environment should also be determined. This offers the best opportunity to define the benefits of migration and to help narrowing down migration strategy options.

1B - Characterization of target environment

When planning a migration project it is also important to understand the capabilities and architecture of the target technological environment. Knowing what users want from the new repository (or disliked about the old one) and understanding its architecture will guide the development of the data migration routines, including mappings, data selection, time behaviour, etc.

1C - Data analysis

A legacy repository is often comprised of a wide range of distinct information, including structured and unstructured data. In order to migrate the myriad of information into the new repository, it must first be located, examined, defined and delimited.
The aim of the data analysis step is to identify the information sources and information entities that have to be transported into the new system. Information sources include all types of data stored, managed or generated by the legacy system (e.g. digital objects, metadata, logs, user information and configurations).
One may assume that not all data is relevant to be preserved, meaning that some of it can be discarded during the migration process. In order to get a better sense, it is helpful to look at the applications, databases and talk to users to understand exactly what information items are relevant to be migrated. You may find that the overall cost of migration is prohibitive relative to the volume of data that needs to be moved and that a compromise on which data is to be migrated must be done.
Data classification, i.e. the conditions for data access, retention requirements and security measures such as encryption, should also be addressed is this step. Often repositories hold classified information with highly conditioned access. One may have to identify the needs of the IT environment and ways in which data may be segregated and protected from members of the migration team. Even a limited set of classifications will have tremendous impact in the way the migration project may be conducted.

1D - Strategic planning

The objective of the strategic planning step is to identify the business and operational requirements that impact the migration process. Various stakeholders within the institution should be consulted to ensure that their requirements are factored into the migration planning.
This step takes into consideration the information collected in the previous steps and defines the migration strategy to be adopted in the following steps. Migration strategies depend on the size, complexity and business requirements of the repository system. For example:

- **Lite migration scenario** - it typically involves loading data from a single source into a single target. Few changes are required in terms of data quality improvement; mapping is relatively simple as is the application functionality to be enabled. Data integration may be on the back-end of systems and will likely be a once-off, "big bang" procedure, i.e. “one shut system migration” [7, 15].
- **Medium migration scenario** - may involve loading data from a single source into a single target or to multiple systems. Data migration may be performed through multiple iterations, transformation issues may be significant and integration into a common data model is typically complex, i.e. “phased interoperability strategy” [7, 15].
- **Heavy migration scenario** - typically involves providing a solution for application co-existence that allows multiple systems to be run in parallel. The integration framework is formulated so the current repository and future repository can work together, i.e. “parallel operations strategy” [7, 15].

1E - Definition of requirements

All the previous steps enable the project manager to estimate the resources that are needed to perform a successful repository migration. In this step, the project manager will define the success criteria for the overall migration project. These may include service-level agreements, expectations for the new storage infrastructure, and objectives such as reduced management costs, reduced storage expenditures, greater insight into expenditure, a simplified vendor model or greater technical flexibility or stability [4].
In this step the high-level requirements for migration, including the data to be migrated, performance requirements and a contingency plan should be defined. This will be valuable information for the project team and the steps that follow.

Planning & design

The planning and design stage follows the definition of requirements. In this stage the project manager is capable of building a project plan and design all the specifications necessary to drive the development of migration routines and testing procedures.

2A - Project planning

After the analysis & consultation stage we are ready to devise an appropriate project plan. In the plan one should describe the strategy and approach, delineate the scope of migration, define a schedule, identify the necessary resources (human and other types), define technical and business requirements, customer expectations (goals), project deliverables, and create a detailed execution plan.
Creating an effective migration plan is often quite challenging. Different types of data or components may require different migration approaches, and comprise different business and operational requirements, e.g. the downtime window may require creative ways of moving the data.
The migration project plan, which is the end deliverable of the planning phase, will function as the blueprint for the migration implementation.

2B - Design of migration routines

During the data analysis step you have already decided upon which information entities and data sources should be migrated. However, it’s in the design of migration routines phase that the

actual mappings between the legacy semantic elements and the new sematic elements will take place.
A migration project is the perfect opportunity for cleanup. Repository owners are encouraged to shift and sort through information, removing outdated or redundant information, thus reducing the volume of information to be moved. Data cleansing tools can be useful as they allow information to be brought up to current standards and its quality to be measured. However, the effort put into cleansing content should be dependent on the business impact if the content value is incorrect [13].

2C - Design of test plan

After all the analysis activities are concluded we have all the information needed to devise an appropriate test plan. This should entail all the steps necessary to make sure that the migration has met all the requirements previously identified. The test plan should be as complete and specific as possible, e.g. does the new system contain the same number of metadata records of the original system? Have all user-defined attributes been migrated? Are there any encoding issues? Was any file corrupted during the copying process? Do original system invariants still hold in the new model?
The test plan can be implemented entirely by scripts and automatic routines, manually or by a combination of both. In any case, humans ultimately check if the migration has been accomplished successfully, so in practice all test plans end up being a combination of automatic and manual checklists. However, keep in mind that the migrated information has been restructured for the new system and that context has changed, hence it might be difficult to compare with the legacy system.
The testing plan may, of course, be revised during the following stages of the project.

Development

After analysis and planning stages, we have all the necessary elements to begin the development of all migration routines and testing procedures. This stage is where migration tools are actually going to be built (or configured) according to the specifications created in the previous stages.

3A - Development of migration routines

This step is where the migration developers come in and implement the routines previously designed. This may consist of building ETL (Extract, transform and load) jobs, specialized programs or scripts that implement the mappings and specifications created in the design phase. All mappings, quality rules, and field validations should be built into the migration routines.
Keep in mind that one may have to return to this step as many times as necessary to drive migration errors down to zero. Revisiting the development stages for six, seven or eight times is not unheard of [16].

3B - Development of testing routines

This step consists in building the test routines that will validate the success of the migration. The deliverables that come out of this step may include validation checklists, testing scripts or dedicated programs that report any anomaly in the migration process execution.

Setup & testing

The setup and testing stage consists in creating the infrastructure where the target repository system and the migration solution are going to work. For the migration to be effective, one should prepare the infrastructure for full-scale trials of the target system against migrated data. If the specifications are thorough and accurate, this phase should be routine and predictable. A strong technical background and documentation will greatly simplify the provisioning effort [14].

4A - Target environment provisioning

During the target environment provisioning phase, the destination infrastructure and software is prepared for the data transfer. This includes setting up hardware, operating systems, configuring storage volumes, installing the new repository system and configuring it to accommodate migrated information and business rules. Provisioning for a one-to-one mapping is usually simple but for a relayout it may be more complex. However, using information generated from the analysis and design steps, it will be possible to automate many of the provisioning tasks [12].

4B - Rehearsal & testing

After the migration routines have been fully developed and before the definitive migration is executed one should perform a series of migration rehearsals in order to make sure that all the requirements have been correctly implemented by the migration routines.
Rehearsal migrations may be partial or complete. A complete end-to-end migration in the pilot environment is of course desirable. However, depending on the amount of information to be moved, this may not always be possible due to time constraints or even due to the stress that this may cause on the production repository.
After each rehearsal, one should run the entire test plan. The output of this activity will dictate if one can move on to the definitive migration or if we should go back to the drawing board and revise mappings, routines or even the project plan. For example, if testing shows that allowable downtime would probably be exceeded, the migration methodology will need to be revisited [12].
The decisive test is to provide the populated target system to the users that assisted in the analysis and design of the migration project. Invariably, users will begin to identify historical data elements that must be migrated that were not apparent to them during the analysis/design sessions [6].

Execution

If the full-cycle migration trials run without errors, one may move on to the execution stage. This stage consists of the execution of the previously developed migration routines and migrate all the information from the legacy system onto the new infrastructure.
This is where all the effort invested so far is going to be put into practice and any glitch in the process might mean that the contingency plan will have to be employed.

5A - Execution of migration routines

The execution step consists of running the migration routines developed in step 3A. This will be the “actual” migration.
Before proceeding with the migration, it is important to review all the guidance and best practices of the previous steps. Ensuring that the objectives are being met and contingency plans are in place

[13]. Additionally, it is important to keep in mind that all the data ingested during the rehearsal steps must be erased before the final migration.

During execution one should have the contingency plan ready to be used in case the migration execution does not run as expected. Keeping a running copy of the original system ready in case one needs to go back is always a good strategy. It is also common practice to keep the migration team ready for action in case anything goes wrong.

Validation

After the full migration, a complete run of tests should be executed on the target platform. This will ensure that the process has run according to plan and that no errors have taken place. This stage also includes the creation of reporting materials to document the overall process and the cut-over to the new system.

6A - Execution of testing routines

As in the migration rehearsal phase, this step consists of rerunning the entire test plan against the new populated system to make sure that everything went according to plan.

If any inconsistency is detected either by the testing routines or users, the contingency plan might have to be put to practice and a new migration run after fixing the uncovered issues.

In some cases, quick fixes can be made on the running system without having to go through a completely new migration. One might just re-import some data without having to reboot the whole migration process.

6B - Reporting

The reporting step is run side by side with the execution of testing routines. This step basically consists of collecting all the evidences and reports produced by the testing routines in order to document and finalize the validation phase. This constitutes proof that the migration was a success and may very well prevent future legal annoyances or disputes.

An additional step is to save and archive all the migration routines. Data migration is often a one-time exercise, however, with the right tools, protocols and mappings, migration routines can be reused in future projects within the organization or in other clients. A documented report of the migration process will serve as a repeatable reference guide and may also help to diagnose and fix post-migration issues [13].

6C - Cut-over

Once the target repository has been built up and all the legacy information has been migrated, the new system is then ready to run. There are mainly three different strategies to accomplish the transition [3]:

1. The cut-and-run strategy consists of switching off the legacy repository system and start using the new feature-rich replacement;
2. In a phased interoperability strategy, the cut-over is performed in small, incremental steps, each of these replacing a few components (applications or data) of the legacy system;
3. In the parallel operations strategy, the legacy repository and the target system operate simultaneously, with both systems performing all operations. During this period, the target system is continually tested; once it is fully trusted, the legacy system is retired.

The cut-and-run strategy is, in many cases, idealistic because of the risk of cutting over to the new system in a single step putting the whole organization's information flow in an untested and thus untrusted system. On the other hand, phased interoperability is potentially highly complex. To be successful, this method requires the migration team to split legacy system applications into functionally separate modules or to separate the data into portions that can be independently migrated. The monolithic and unstructured nature of most legacy systems makes such an approach difficult, if not impossible. A concrete transition strategy for a particular migration project would probably involve a combination of these approaches, applied to different repository components [3].

Wrap-up

After the new repository has gone into production, there are a few activities that one should consider. These include training users and repository managers to use the new system, collect, build and archive all the project documentation and deliverables, and provide maintenance and support to new system in case of an emergency or if any tuning is necessary.

7A - Training

No system adoption is complete without training of its end-users. Through their insightful questions, you will quickly learn how the target system should be reconfigured or enhanced, both crucial inputs for this and future migration projects.

As training is known for having a short lifespan, it is normal to postpone training until the end of the project. However, training key end-users may be done earlier in the project to assist in the configuration of the system [14].

7B - Documenting

After the migration has been completed, the project team should compile all the migration statistics, reports, designs and plans and prepare a report to highlight what worked, what didn't and lessons learned. The report should be shared with all members of the migration team. These reports are critical in building a repeatable and consistent process through continuous improvement [7].

7C - Supporting

The supporting phase consists of keeping a team of technicians ready to assist users with any question or operational difficulty.

Post-migration issues may be of informational nature (e.g. information missing, bad mappings, etc.) - which, in this phase, can usually be fixed directly on the production system – or system nature (e.g. bad configuration, bad tuning, among others).

Conclusions and future work

Repository migration is an inevitable process that any institution that hosts or manages a digital repository will have to go through. It's just a matter of time for the running repository to become incompatible with current technologies, inadequate to serve the business needs of the institution or simply old-fashioned, not meeting the expectations of its designated community.

Although a complex and risky process, with the proper preparation and guidance, a repository migration its definitely a endeavor that is worth the investment.

A comprehensive methodology, a well prepared team and a clearly defined project plan is always a good recipe for success. Even so, it often causes major disruptions as a result of downtime or performance issues, which can have negative impact in users perception of system quality and future productivity.

To prevent these problems, organizations need a consistent and reliable methodology that enables them to analyze, plan, design, develop, migrate and validate the migration. Potential pitfalls can be avoided by following the best practices presented in this paper.

Future work is focused on enhancing these guidelines with implementation recommendations and examples of practice. The guidelines will then be published in the form of a technical whitepaper for a more insightful reading and to reach a wider audience (to be published as a SCAPE project deliverable).

Also, a quick consultation checklist is being created to aid IT professionals to double check that all the angles have been covered during the preparation, execution and post-migration stages of a repository migration project.

Acknowledgments

Part of this work was supported by the European Union in the 7th Framework Program, IST, through the SCAPE project, Contract 270137.

References

[1] Bearing Point Inc. 2008. Data Migration through an Information Development Approach - A management overview.

[2] Bisbal, J. et al. 1999. Legacy information systems: issues and directions. IEEE Software. 16, 5 (1999), 103-111.

[3] Brodie, M.L. and Stonebraker, M. 1995. Migrating legacy systems: gateways, interfaces & the incremental approach. Morgan Kaufmann Publishers Inc.

[4] Burry, C. and Mancusi, D. 2004. How to plan for data migration. Computer World.

[5] Harris, L. 2010. IPM 11g Migration Best Practices. Nexus'10 (2010).

[6] Hudicka, J.R. 1998. An Overview of Data Migration Methodology. Select Magazine. (1998), 5.

[7] IBM Global Technology Services 2007. Best practices for data migration - Methodologies for planning, designing, migrating and validating data migration.

[8] Jantz, R. 2005. Digital Preservation: Architecture and Technology for Trusted Digital Repositories. DLib Magazine. (2005), 1-17.

[9] Lawless, D. et al. 1997. The Butterfly Methodology□: A Gateway-free Approach for Migrating Legacy Information Systems. Proceedings Third IEEE International Conference on Engineering of Complex Computer Systems Cat No97TB100168. (1997), 200-205.

[10] De Lucia, A. et al. 2008. Developing legacy system migration methods and tools for technology transfer. Software Practice and Experience. 38, 13 (2008), 1333-1364.

[11] Mohanty, S. 2004. Data Migration Strategies, Part 1. Information Management.

[12] Network Appliance Inc. 2006. Data migration best practices.

[13] Open Text Corporation 2009. Top 10 best practices in content migration.

[14] Pick, B.G. 2001. Data Migration Concepts & Challenges.

[15] Rahgozar, M. and Oroumchian, F. 2003. An effective strategy for legacy systems evolution. Journal of Software Maintenance and Evolution Research and Practice. 15, 5 (2003), 325-344.

[16] Utopia Inc. 2009. Data migration management - A methodology: Sustaining data integrity after the go live and beyond.

Author Biography

Miguel Ferreira has a Ph.D. by the University of Minho in Technologies and Information Systems with a specialization in Digital Preservation. He is a founding member of KEEP SOLUTIONS. Prior to joining KEEP SOLUTIONS, has worked as a researcher at the University of Minho, consultant at the OPorto Regional Archives and as a researcher for Philips Research.

Luís Faria has a degree in Systems Engineering and Computer Science by the University of Minho. Prior to joining KEEP SOLUTIONS, has worked as a researcher at the Portuguese National Archives, and at CERN – European Organization for Nuclear Research.

José Carlos Ramalho has a Ph.D. by the University of Minho in Programming Technology. He is a founding member of KEEP SOLUTIONS, assistant professor at the Department of Informatics of the University of Minho and a researcher of the Computer Science and Technology Center. Prior to joining KEEP SOLUTIONS, has worked as a freelancer programmer, and as a networks engineer.

Table 1 - Unified methodology for legacy repository migration.

Stage	Activity	[5]	[6]	[14]	**[1]**	[4]	[11]	[12]	[7]	[16]	[13]	[10]	[9]	[3]
1 Analysis & consultation	**1A** Characterization of legacy environment		□			□			□	□	□	□		□
	1B Characterization of target environment										□	□	□	
	1C Data analysis		□			□		□		□	□	□	□	□
	1D Strategic planning		□		□			□		□	□			
	1E Definition of requirements	□		□	□	□			□				□	
2 Planning & design	**2A** Project planning	□		□	□		□	□	□	□				
	2B Design of migration routines	□	□		□	□	□	□	□		□	□	□	□
	2C Design test plan								□		□			
3 Development	**3A** Development of migration routines	□		□	□				□	□		□	□	
	3B Development of testing routines													
4 Setup and testing	**4A** Target environment provisioning				□			□	□			□	□	□
	4B Rehearsal & testing		□				□	□	□	□		□	□	
5 Execute	**5A** Execution of migration routines	□		□	□		□	□	□	□				
6 Validate	**6A** Execution of testing routines	□	□		□	□	□	□	□		□	□	□	□
	6B Reporting								□		□			
	6C Cut-over	□		□	□				□	□		□	□	
7 Wrap up	**7A** Training													
	7B Documenting				□			□	□			□	□	□
	7C Supporting		□				□	□	□	□		□	□	