Data Migration

A White Paper by Bloor Research

Author : Philip Howard

Publish date : September 2007

In our view, data migration has historically been under-valued, under-resourced and not treated with the attention it deserves

Philip Howard

This document is sponsored by:

According to research conducted by Bloor Research (see *http://www.bloor-research.com/research/survey/876/data_migration_survey.html*) industry currently spends in excess of $5bn per annum on data migration, including software, services, consulting and so forth. Yet there is no market for data migration per se. Nor is it recognised as an independent discipline within data management. Indeed, the tasks required in data migration are typically treated as being just a part of what you need to do when migrating, implementing or upgrading application software.

These application projects frequently overrun their budgets, get delayed or, in extreme circumstances, get cancelled. We believe that a major reason behind these failures is precisely because the techniques and disciplines of data migration are not treated seriously enough or are not well enough understood. This is supported by our research, which indicates that more than 60% of data migration projects have overruns on time and/or budget.

In our view it is time to reverse this situation. It is time for data migration to be recognised as a discipline it is own right and for its exponents to get due credit when their job is well done. It is time for data migration to be treated as a major component, with its own requirements, within larger application projects rather than as an afterthought.

This paper is not supposed to be a how-to guide for data migration. Such can be found in John Morris' "Practical Data Migration" published by the British Computer Society. What it is intended to do is to highlight the major issues involved in data migration. We hope that an understanding of the complexities involved will help readers to understand why we believe that data migration, and the people who do it, need to have their status elevated. At the least, we hope that the issues involved in data migration will be taken more seriously, especially in the planning stages of broader application projects.

According to the Wikipedia, data migration is "*the transferring of data between storage types, formats or computer systems… it is required when organizations or individuals change computer systems or upgrade to new systems." According to the Webopedia it is 1) the process of translating data from one format to another… necessary when an organization decides to use a new computing system or database management system that is incompatible with the current system and/or 2) the process of moving data from one storage device to another, in which case data migration is the same as Hierarchical Storage Management (HSM)."*

In part because of the qualifications ("*it is required…*", "*necessary when…*") that form both of these definitions, neither is correct. For example, the Wikipedia definition implies that if you upgrade your system then you will need to migrate your data, which isn't true. Similarly, you might infer from the Webopedia definition that compatibility is a key issue. Indeed it is but this definition might suggest that existing systems are compatible when they are not.

Migration literally means movement from one place to another. However, if you were to move from Boston to New York or from Manchester to London one would not normally describe that as migration. On the other hand, if you moved (on a permanent basis) from Boston to Paris then you would be (e)migrating from your perspective or (im)migrating from the point of view of a French person. The difference is that you are moving from one cultural, linguistic and political milieu to another, which will involve significant change. This is the essential difference between moving and migrating: the latter involves major change while the former does not.

To take a simple example, if you wish to consolidate multiple (Oracle) Siebel CRM systems to a single instance of Siebel, and all of those systems use the same version of that software (and the systems have not been customised, which admittedly is unlikely), then this would represent a data movement task and not data migration. Conversely, consolidating multiple, heterogeneous CRM systems into one new one would data require data migration. The difference is that the structure of the data will be different across heterogeneous systems and that data will need to be altered in some way (including format and style changes, semantic reconciliation, de-duplication, perhaps just that the data is now used differently, and so on) in order to fit the demands of the new system.

Similarly, moving data from one storage device to another, for example, is data movement: data migration would only be involved when the format of the data has to be changed in the process. For the same reason, if you move your applications from an Oracle database to MySQL, say, but retain your existing applications (and therefore, the same schema) then that is a data movement exercise and not one of data migration.

A further consideration is with respect to conventional ETL (extract, transform and load) processes. Under both of the definitions quoted above, ETL, at least as it relates to loading data into a data warehouse, would be considered a form of data migration. Certainly some of the same techniques might be employed but we do not believe that this is how data migration is commonly understood. So, we need a definition of data migration that excludes both pure data movement and the sort of ETL processes that are used to load a data warehouse.

Finally, a further distinction must be made between data migration and data integration more generally. The former is a one-time exercise whereas most other forms of data integration, such as loading a data warehouse, are on-going.

It is worth considering a number of common use cases to see whether these actually involve data migration or not:

- Moving from one or more databases or storage types to another, with all other things remaining constant – not data migration.
- Loading a data warehouse – not data migration.
- Archival, where older or less used information is moved to secondary storage – not data migration.
- Migrating from one application to another (such as Oracle to SAP) – will require data migration.
- Implementing a new application where (some of) the data for the new application already exists within existing applications – will require data migration.
- Upgrading from one version of an application to another – may require data migration, if data formats change.
- Implementing master data management – may require data migration, if there is a physical movement of the data into, say, a hub. In other circumstances, procedures similar to conventional uses of ETL may be needed.
- The capture, transformation and pass-through of data taken from message queues; EDI, SWIFT or HIPAA messages; or via change data capture – does not involve data migration.

The first conclusion we can draw from this is that data migration never exists in isolation: it is always linked to applications in some way, whether that be an application migration, implementation or upgrade. This is one reason why there is no such thing as a data migration discipline per se, because it is always a subset of some broader effort. However, this may be a reason but it is not an excuse: the use of ETL is part of a broader discipline but it does have its own discipline area.

Getting back specifically to the question of how we define data migration, the second conclusion we can draw from the use cases above is that the usage of the data, if by that we mean whether the data is used for operational or analytic purposes, remains constant. With data migration you never migrate from an operational environment to an analytic (business intelligence) environment or vice versa.

Note that we are defining usage here as distinct from context. Loading data into a warehouse changes the context in which it is used, as does migrating data from an ERP system into a CRM system. However, not all data migrations change context. For example, the consolidation of heterogeneous CRM systems does not change the context of the data: it is still CRM.

Our definition is therefore as follows:

"Data migration is any movement of persistent data that involves some sort of restructuring of that data while the usage of that data (operational versus analytical) remains constant."

Note that you could optionally replace "*movement of persistent data*" in this definition with "*one-time movement of data*". Either of these qualifiers will exclude the sort of data integration that involves the transformation of things such as SWIFT and EDI messages, which is an on-going process for what are essentially transitory objects (though these may be persisted this is done for auditing reasons rather than operational processing; in other words the usage would change).

It is also arguable that there are two classes of data migration: simple and complex, where the former retains the same context (that is, the same application type) while the latter involves a new context (a new application type or, perhaps, MDM). While we will not be pursuing this idea further in this paper it is worth bearing in mind this additional complexity.

It is our belief that a major part of the reason behind the failure of so many migration projects to meet target and budget is because of the lack of regard for the expertise and processes required for data migration. In this section we will examine the reasons behind this in more detail.

The implication of applications

Data migration is invariably a function associated with applications. More precisely, data migration is always a subset of some overarching application project. The most important implication of this is that it is always those who are making decisions about the application(s) that are in control of the project. Unfortunately, such individuals tend not to be experts in dealing with data: focusing on processes and interfaces rather than data semantics. Often these people will not appreciate the scope of any data issues that may be involved in the project and they will frequently underestimate what is needed in terms of time, budget and personnel in order to manage those issues. This is exacerbated by the fact that data migration is not recognised as a discipline in its own right within data management so there are few, if any, in-company experts who can advise on such matters. Moreover, it is commonly the case that systems integrators will separate the application migration aspects of a project from the data migration component (leaving this to the customer), ostensibly because they do not understand the data as well as their client but, at least in part, because they recognise its difficulty.

What all of this means is that, if not actually an afterthought, the issues arising from data migration are not given the full consideration they need in many projects. This frequently results in cost overruns, late delivery and lack of user acceptance for the overall project when it is finally delivered (if it ever is). In our view, it is the failure to give due consideration to data migration that is the most common cause of these problems.

What is needed is a strong data management presence within the overall project team, so that data issues are given proper consideration. Further, the data management team members must understand the techniques and strategies required for data migration, along with the tools that can be used to assist with those processes. However, this raises issues of its own.

The prevailing view of data migration

Within most organisations there is no such thing as a data migration expert. Indeed, data migration is regarded as a dead-end job that nobody wants. Why? Well, to begin with you get roasted if you fail but you don't get any kudos if you succeed. Secondly, it is regarded as boring. That's a matter of opinion. Thirdly, it is a short term job. You are dedicated to the task for 6 or 12 months but what are you qualified to do afterwards? Fourth, your team will be seconded out to the applications guys whenever there is a problem over delivery on the software side (because those are the guys that make the decisions, remember, and therefore what they are doing is more important), which means that you can't properly plan your part of the project. Fifth, as soon as the project is finished you will be out of a job, which is hardly an encouragement to finish on time. Sixth, if the greater project is an application migration project (from Oracle to SAP say) then you will be working mostly with the Oracle experts who themselves will be out of a job when the project is done. Finally, such projects can often become embroiled in what seem like intractable political struggles and why would you want the stress?

Needless to say, everybody in the IT department knows all of this (at least viscerally if not in detail). As a result no-one wants to work on the data migration team. This in turn means that, in many cases, it is the less experienced people and the staff with less political clout that get the data migration jobs. So, they are even less well placed to ensure that the necessary resources are available to the team. Is it surprising that so many application migrations and upgrades run over time and budget? For very many companies this is almost inevitable before the project even begins, because not enough emphasis is placed on data migration.

We do not think we are saying anything new here. However, the question that then arises is why matters continue in this fashion and why people continue to make the same mistakes? Of course, the simple answer is that this is a result of organisational staffing and budgetary practices but that is not very helpful as an explanation. In fact, we believe that the heart of the problem lies in the fact that the people who manage the data in most IT departments are regarded as second-class citizens: applications are sexy, data is not. Moreover, even within the data management group, data migration is about the least sexy thing you can be doing.

In other words, the root cause of data migration problems derive from a perception of data management in general, and those involved in data migration in particular. In our view, this perception (and especially with respect to data analysis—see later) needs a wind shift of 180o. Companies that continue to have the sort of view of data migration just described, will continue to have failed and over-running projects.

Using external resources

Recognising the relative importance of data migration by an individual is one thing but changing the culture of an organisation is another. It may well be that members of the data management team recognise the importance of these arguments as will, perhaps, the CIO and some others but if the IT department as a whole is not convinced then problems will persist. In such environments it may be better to consider the use of external specialists for data migration rather than use in-house resources. While consultants approach these matters in different ways (some will not touch data migration), the approach that we would recommend would be one where the external data migration specialist charges a fixed fee for an initial evaluation after which they will provide a fixed price quotation for doing the job as a whole. It will be an advantage if the consultant can provide hosting services on a temporary basis during the cut-over process (but depending on the data migration strategy adopted—see later).

Note that data migration consultants can only do what they are asked to do. Give them the data and tell them what transformations are required and they should be able to migrate it for you. However, as we shall see in the next section, herein lies the rub: you often don't know where all the data is or how you will need to transform it.

One final point—as we discuss later, data migration is essentially a business issue—this means that whatever external resources you use, you should not give them ownership of the project.

There are many aspects of data migration projects that are similar to those that are required for other types of project, such as project control itself, documentation and reporting, testing, the building of data models and so forth. However, there are also requirements that are much more specific, albeit that there is still overlap with other types of migration and data movement projects. In this section we shall consider these specific requirements in terms of the techniques that are required and the strategy that might be adopted.

There are seven broad techniques that need to be used:

1. discovering and selecting your sources,
2. understanding your data,
3. cleansing your data,
4. transforming your data,
5. moving your data,
6. testing and validation,
7. auditing and documentation.

The last three of these are fundamental to any successful project but they are, we believe, well understood (though testing, which is probably the least sexy part of the non-sexy data migration, can be problematic because you have nothing to compare your results with since you have, by definition, a new context for that data). Moreover, they are not typically the cause of data migration problems. Thus this section focuses on the first four of these techniques.

Source discovery

There is no technique or methodology for source discovery per se. However, that does not mean that it is not important. The point is this: you need to know about all the sources of data that will populate the new system. To take a very simple example, suppose that you are migrating from SAP to Oracle. It might appear that the answer is straightforward: the database(s) used to host SAP will source the Oracle application. However, it may well be that there was a separate HR system that will now be incorporated within the new system. Okay, you probably know about that. What you may not know about, and what even your IT colleagues on the application side may not know about, is that the Finance department (say) want to use the new system because it expands on the capabilities that were previously available, which they had previously had to provide through Excel spreadsheets and local Access databases.

Along with the identification of data sources there is also an issue with respect to selecting the most appropriate data source to use when data can be found in more than one place. A common mistake here is to assume that the "corporate" data source is the most accurate one. In practice, it may well be that a salesman's personal records have better contact details than the company's CRM system, or a maintenance engineer's notes may be more comprehensive than the asset inventory.

The first step in identifying the scale of the data migration sub-project must therefore be to ensure that you know about all the data sources that will feed into the new system. Unfortunately there is no tool to help you do this; all you can do is to talk to your users. Note that we don't believe that talking to your colleagues in the IT department will be sufficient: they probably don't even know that these Access databases exist. In any case, as John Morris states in his book "Practical Data Migration" (published by BCS) "data migration is a business not a technical issue" and "the business knows best" so you cannot expect to run a successful data migration exercise without talking to users. Moreover, it will be the business users that sign-off the project at the end of the day so keeping a close liaison between the project team and the business should be a major consideration at all times. It is also worth noting that users can often be reluctant to sign-off on projects so keeping them fully in the loop is of paramount importance.

These two quotes represent John's first two Golden Rules of data migration. Bear in mind, though, that the business won't know all you need to know: migration requires the business and IT to work in collaboration.

Once you know where all of your data sources are, you can identify all of the data that you have that can be used to populate the database for the new system. At this point you need to do a gap analysis: does the data that you have match the requirements for data in the new system? In particular, are there any mandatory fields in the new application for which you do not currently hold data? If the answers to these questions are no then you have either missed a data source or you genuinely do not record that information at present, in which case you will have to decide how to collect that data.

On this note there is a further point: what about data that you don't know that you know about? For example, you may not think that you know someone's postal or zip code but you can use enrichment tools to automatically discover this information. Similarly, you can infer other sorts of 'missing' data by using profiling and other tools. However, this is the subject (at least in part) of the next section.

Understanding your data

Once you have identified all of the sources that will be used to populate your new system you need to understand which elements from each of these sources are required. Moreover, you need to be able to recognise any errors or omissions in that data, as improving the quality of that data (see next section) is invariably a part of the data migration process. In addition, once you have identified the errors in your data, and any inconsistencies or discrepancies that exist across data sources, you can start to estimate the time required (and costs involved) to bring this data up to standard.

Understanding your data involves investigating it at various levels of granularity:

- At the field level: for example is each data item appropriate for the field it is currently in, and is it of the correct type (numeric, alphabetic and so on) and appropriateness for its destination field?
- At the table level: for example, what are the implications for the source, for transformations and for the target of any primary/foreign key relationships?
- At the cross-table level: are there any implied relationships, such as account structures or product hierarchies that must be taken into consideration either in the analysis of data sources or in the process of conversion?

These are the sort of questions that data profiling and analysis tools can answer as well as other capabilities such as the identification of redundant fields, for example. The problem, of course, is that many of these relationships may be implicit, or have been written into applications rather than being explicitly managed by the database. Moreover, particularly in the case of older systems, relationships may be undocumented and forgotten. Further, complex relationships may not be described in the metadata so that any relationships that exist must be inferred directly from the data.

Why are these relationships important? Because they may represent business rules that need to be incorporated within the new system. In this respect, technical relationships are not so important: you would expect to implement referential integrity (for example), at least (in theory) in the design phase, but if you are not aware of an implicit business rule then you certainly won't. However, it is not simply a question of inferring business rules and then implementing them. Some of these rules may be out-of-date and no longer required while others may be replaced in the new system. How do you find out? You will have to talk to the business users again.

Finally, what we have not discussed is the understanding of data relationships across data sources. How does the data in my Oracle database relate to the data in my DB2 database, for example? Unfortunately, most data profiling tools cannot compare data across multiple data sources, simply because they weren't designed to do this, which means that this effort tends to be manual, tedious and expensive. Some relationships, of course, are simple: there will be a one-to-one relationship between customer name fields across different systems, for example. Nevertheless, manual examination will miss deeper and more complex relationships that may not be immediately obvious. Fortunately, there are now a few products that do have 'data relationship discovery' capabilities so it will be worthwhile considering the use of tools that can automatically discover these for you.

Cleansing your data

While the techniques of data cleansing are well understood, here we want to discuss their application. This is because bringing the data up to standard does not mean seeking for perfection in data quality. John Morris' Golden Rule 3 states that "no organisation needs, wants or will pay for perfect data quality". We agree, and for several reasons. First of all, you cannot practically get perfect data quality even at a single point in time. For example, you have two records for customers named J Smith with the same address and telephone number. Is this a single person or are they father and son who live together, or two brothers, or is it merely a coincidence? Without further information you cannot possibly know. You might choose to consolidate those customers' records into one and you might be right 19 times out of 20 but you won't even know on which occasion you have got it wrong.

The second problem with perfect data quality is that data quality deteriorates. The generally accepted figure for this is between 20 and 25% per annum or approximately 2% per month. If we take customer data, for example, according to US census records 14% of people move each year, .8% die, .8% get married and .4% get divorced: thus a total of 16%. Depending on your business you probably won't get told about this when it happens, which will mean that your data is at least partially inaccurate. Now add in all the various causes of errors such applications that are out-of-date with respect to business processes (see box later in section) as well as operators and users making typos or leaving blank fields where data entry should have been mandated, or where default values are entered rather than real ones, and so on. Since all data entry is driven by people and we are, by nature, error-prone, even if you start with perfection it will soon deteriorate. Having said that, you can put in place real-time data quality routines that will check entered data

As an example of an application and a business process not being in step, one utility company had codes like "LDIY" and "KUM" inserted into name fields, these being directions for meter readers meaning "large dog in yard" or "key under mat". Needless to say there was a definite negative impact when these codes were included in mailings. The problem was that the application was out-of-date with respect to the business processes it was supposed to be supporting and had no field to enter these codes. This happens regularly both because of poor application design and because the pace of updating business processes is faster than that for updating applications.

and query it if something odd is found. However, even that cannot prevent people moving house, much less typos and some other types of errors so that 100% data quality can never be assured.

So how good should the data quality be? If you ask this question, the sort of answers you will get will be along the lines of: "pretty good", "very good", "excellent" and, of course, the impossible "perfect". This isn't a lot of use: what you actually need are some measures: 98% accurate or 95% or whatever. Fortunately, there are data quality tools that provide dashboards and metrics that do allow you to monitor data quality rates (and which are also useful for data governance purposes—but that's another story). However, this isn't as simple as it seems because data quality really needs to be considered in multiple parts. This is because some data may be required to make the new system run or run properly. For example, if there is a mandatory field in your application and you have records that have no values for that field then the software won't run. And what happens if you have a value but it is the wrong one? This may or may not be critical. Conversely, there may be non-mandatory fields where incorrect data has a major impact on results or, on the other hand, has very little impact at all. And then, as a separate exercise, there is the question of the impact on the business if this data value is incorrect. It should be clear that the quality measures you require, and the effort you need to put into data quality, will be different depending on the answers to these questions. Moreover, there is also a question of context,; for example a loading dock might well have a valid address, which would be useful as a shipping address but useless as a contact address.

The discussion points in this section are by no means comprehensive and there are many other issues to consider. For example, what about the currency of your data? Are codes used correctly and are they still meaningful? Do you have language conversion issues? What about levels of granularity—is a "customer" a person at a particular address, a person who may have multiple addresses, all the people at a particular address? And so on.

Transforming your data

There are two relevant aspects to the transformation of data: determining what transformations you need and then the actual process of implementing those transformations. In many data migrations, especially complex ones, the former is much harder than the latter. Unfortunately, we know of no tools that can help you with this sort of discovery: it is simply a manual exercise that needs to be accomplished once you have a full understanding of both your source and target systems.

In so far as implementing those transformations, the actual process of first mapping and then transforming your data are well understood once the discovery phase has been completed and we will not discuss these in detail. However, there is one point that we will make, which is to avoid one-way transformations if at all possible. That is, a transformation that you cannot reverse. For example, if you map field A to field B then it is relatively simple to reverse this mapping but it is entirely possible to create mappings, (usually when mapping multiple fields into one target field though not all such mappings create problems: for example, a concatenation is fairly easily reversed) that you cannot create a reverse transformation for. The point here is that if this part of the migration does not work for some reason and what you really want to do is to back out to some previous point in time, then you will be up the proverbial creek if you have transformations that you cannot back out of.

Unfortunately, it is not always possible to avoid such mappings so extra care should be taken when these are employed. Best practice would be to be really careful about your testing of such transformations and to make sure that you always keep a copy of the original data around in case you need to go back and re-do something. Having such a copy around is also critical in testing, training and may even be needed in the future if somebody discovers something unexpected and you need to go back and do a detailed data lineage investigation.

Quality assurance of transformations is also an issue: since you have migrated the data it has been restructured, so you cannot compare the old with the new as you want them to be different. So, what do you compare the migrated data to, to ensure that it was migrated properly? In practice, there is no simple answer to this. You can compare it with expectations but it is probably only user acceptance testing that will determine the success or otherwise of the migrated data.

Applying these techniques

Finally, we should consider how these techniques are applied. While we have listed them in 'chronological' order we do not believe that they should be implemented using a conventional waterfall approach. The danger then becomes that you never get to the end of the 'understanding the data' phase. Moreover, if you have a project plan that reads discovery: months 1–4, movement: months 5–7, test and validate: months 8–12 you can easily end up not knowing until month 9 that you have a project slippage.

While you certainly need to do a detailed profiling overview in order to set timescales and budgets, during the actual process of the work a more flexible and iterative approach is required, whereby you profile each subset of the data in turn, cleanse it, transform it and test it. For example, if you are doing a SAP migration then you might start with the materials master, move that data, validate that all of the data from the source has been properly moved, then move up the chain to the next set of data that depends on that lower set, and so on. In other words, you need to take an 'agile' programming approach in which of these techniques is used, in turn, in an iterative fashion. This way you may find that you have a project slippage in week 6 but it is better to know that earlier rather than later.

Note that while profiling and discovery is an iterative process it generally needs to be methodology driven. One approach is to perform a 'target led analysis', focusing the profiling on entities and attributes that are most important to the target system (for example, key entities, primary/foreign key fields, dependencies and business constraints). There is little point in performing analysis on attributes that are not going to be migrated so what you need is a cut and dice approach, along with a well-defined methodology, particularly for dealing with data quality problems as they occur. Data migration teams should work with the business analysts during the profiling process to resolve data quality and business data problems as they are discovered, rather than leaving this until late in the project lifecycle.

Allocating resources

There is one other problem associated with data migration that we also need to mention and this is specifically with respect to the budgeting of project teams. Because of the agile and iterative nature of migration projects you will typically require short bursts of specific skill set resources at various times during the project. What this means is that the percentage utilisation of your resources is a major factor in cost control. For example, if the total project requires 1,000 man days of resource but you can only utilise those resources at a rate of 75% then your budget will need to reflect this fact unless those resources can be re-deployed. However, the danger is that upon re-deployment those resources will not then be available for the migration project when they are needed, which will impinge on project timescales. And, of course, if the company changes its priorities at any time during the course of the project then utilisation can easily drop to 50% or lower, bringing about a consequential cost increase.

What is needed is an approach that is predicated upon pooled resources or very flexible contracting teams. In large enterprises, where multiple migrations may be taking place (our research indicates that the average Global 200 company runs 4.48 migrations per annum) it will make sense to run these migrations in parallel so that resources can be shared on a staggered timeline.

Whereas you need all of the techniques of data migration, the strategies you might apply to data migration and similar projects represent options. There are essentially three of these: big bang, parallel running and incremental, which we will discuss in turn.

Big bang

Using the big bang strategy you continue running on your original system(s) throughout the duration of the migration project. The new system is not up and running at any stage. At more or less the last moment you load all of your data onto your new system, turn off the old one and then start running in the new environment. The two big advantages of this approach are, first, that it eliminates the problems of having dependencies between systems where using non-big bang approaches some of the old systems have to run side-by-side with the new ones; and second, this approach minimises your hardware and systems requirement since the only time during which you have two systems running is during the changeover period. This, at least, is the theory, though it is often not realised in practice.

This changeover process typically takes a weekend though we know of one telecommunications company that turned off its billing system for two weeks during such a changeover. Consider the cost of that to the business. No doubt the company originally expected that it could do this over the proverbial weekend but in the event it turned out to be a much bigger bang than was originally anticipated. However, even if the weekend is all you need we are increasingly living in a world where you cannot take part of the business down at all. It may be fine for an HR application, say, but for any on-line applications the cost to the business is likely to overwhelm any savings to be made by taking this approach.

The second downside of this strategy is what happens if the new system doesn't work or doesn't work well enough? If this happens then either you extend the length of the weekend (with all the costs to the business that that implies) or you go back to the original system and try against next weekend and/or the weekend after and so on. In either case you will do serious damage to the reputation of the IT department with the user community and you are also starting to lose the cost advantages of not having to run two systems at the same time. Moreover, there are costs to the business itself, since there may be a direct customer impact, new product launches may have to be delayed, and so on, which can have a big impact on both the top and bottom line.

Parallel running

Parallel running starts the same way as the big bang approach but you do not intend to cut over to the new system once the data is loaded. Instead, you will run both systems in parallel, with the original system(s) continuing to serve the business while the new system is thoroughly tested until you are ready for a final cutover. The advantage of this approach is that the new system is more or less guaranteed to work correctly when you finally move to it. In addition, you should have minimal downtime between turning off the old system and turning on the new one. The downside is that this parallel running is expensive, not just in computing resources but also in terms of the people required to run it, and the checking that needs to go on to ensure that the new system is doing what it is supposed to.

Parallel running is also more complex, as an environment, when compared to the big bang approach. This is because incoming data is entered into the old system and you also need to propagate that data to the new system. This means that you need some sort of synchronisation software to keep the two systems in parallel.

Incremental migration

Incremental migration works in a similar fashion to parallel running except that you incrementally turn on the new system rather than turning it on all at once. So, for example, you might be in a position where transactions for some customers are processed by the old system and transactions for others are processed by the new system. Similarly, some functions might be executed by the old system and some functions by the new one, though this will be as a result of the new application(s) rather than data migration per se. This is one of the advantages claimed for incremental migration: that you can start to a get a return on your investment much sooner.

Note that by the time that everything has been moved it will be, more or less, time to turn off the old system, so there will be very little, if any full-scale parallel running.

The traditional method of supporting incremental migration is to place a flag against the relevant data fields in the source database, which tells the application to look for that data in the new system. The disadvantages of this approach are that you have to amend all of the applications that address this data and you have to amend the data tables (and the database schema) in order to enable this, and performance will deteriorate when the data is in the new system. An alternative approach is to use a state model that keeps a record of the current state of the data migration (that is, it knows which data is live in which system). The application is then directed initially at the state model, which tells the application where the relevant data is located. Note that there is no requirement to amend any data sources when using this method and for this reason alone the state model approach is to be preferred over the flag-based method. However, this is not the only reason why it is better.

Whatever strategy you adopt for data migration it is important to be able to back out of actions that do not work. However, this is relatively difficult to do when using big bang or parallel processing strategies because in both cases you are moving whole sets of data in a very short period of time. When using incremental processing you only move relatively small datasets at a time. This in itself is an advantage since this makes testing easier because it, too, can be done incrementally. Similarly, if you do need to back out of an increment then this is a much easier process because of the size of the dataset. Furthermore, the use of a state model can assist in this process because it maintains details about the state of that data.

Zero-downtime migration

While not a first-line strategy such as those we have just discussed, a secondary strategy is whether you wish to adopt an approach predicated on having zero downtime (or as near zero as possible—we know of one implementation where, after a nine month migration project, there was 4 minutes of downtime—but this was because the company was changing its Application Server) during the migration. This decision will depend on the importance of the application and data you are migrating. If this supports a web-based application or some other mission critical application that needs to be up 24x7 then a zero downtime approach should, properly, be mandated; though this may not be the case if you already use mirrored, redundant systems where you can turn one of these off for the duration of the migration and leave the other one(s) running. Zero-downtime migrations should also be considered whenever a failure to go live, accurately, on time, may be detrimental to staff morale.

Note that a zero-downtime approach is more or less implicit within the incremental and parallel strategies but is inimical to the big bang.

We recommend the following actions:

1. The data migration aspect of the project should be set up with its own management team and its own project planning. It should be treated as an independent sub-project within the overall project, with its own budget and resources.

2. The data migration project team needs to identify all the relevant data sources and undertake initial profiling and cross-system data analysis, so that the team can estimate the total costs, resources and time required for the sub-project BEFORE budget and delivery schedules for the whole project are finalised.

3. As a matter of principle, staff should not be re-allocated back from the data migration team to fire fight other issues within the project or elsewhere within IT, unless the data migration sub-project is ahead of schedule, and then only to the extent that is not put behind schedule.

4. We recommend the use of appropriate tools, both in order to understand the data to be migrated and its relationships, as well as for transformation and moving the data. In the case of profiling, data relationship discovery and analysis tools, the environment (especially for any large scale migrations) is likely to be too complex to achieve this manually. In the case of transformation and movement tools we recommend their use primarily because they should make it much easier to back out of failed processes. In addition, the fact that an audit trail and documentation should be automatically provided will be an advantage. We also recommend the use of data cleansing tools, not just for their own sake but also because they will allow you to measure the quality of data (and they may also be useful for certain types of complex transformations). In addition, if the state model approach for incremental migration is adopted, then an appropriate tool will be required. The same is true if you are adopting an approach that employs data synchronisation in a parallel processing environment.

Note that we have not specifically advocated a change in the status of staff involved in data migration. We do not believe that you can change people's mindsets simply by suggesting that it be so. However, we do think that establishing data migration as a sub-project in its own right will start this process. Moreover, the consultation prior to finalising overall budgeting and timelines will also help in this, as well as ensuring that delivery schedules and costing are realistic, at least as far as the data migration part of the project is concerned.

Conclusion

In our view, data migration has historically been under-valued, under-resourced and not treated with the attention it deserves. As a result, many major migration projects have failed either partially or completely. The remedy for these failures, we believe, is to accord data migration the respect it deserves: to consider it in detail prior to the initiation of relevant projects and not as an afterthought. In turn, this will result in increased status for data migration as a discipline, which will, as a result, mean that more experienced and capable staff are attracted to this function. In a virtuous circle, this will then improve the practice of data migration, which will feedback into improved costing and delivery for broader projects that involve data migration.

Further information

Bloor Research will keep a page dedicated to this white paper on their website for further information on this topic as it becomes available. The page can be accessed at: http://www.bloor-research.com/research/white_paper/875/data_migration.html

Bloor Research overview

Bloor Research has spent the last decade developing what is recognised as Europe's leading independent IT research organisation. With its core research activities underpinning a range of services, from research and consulting to events and publishing, Bloor Research is committed to turning knowledge into client value across all of its products and engagements. Our objectives are:

- Save clients' time by providing comparison and analysis that is clear and succinct.
- Update clients' expertise, enabling them to have a clear understanding of IT issues and facts and validate existing technology strategies.
- Bring an independent perspective, minimising the inherent risks of product selection and decision-making.
- Communicate our visionary perspective of the future of IT.

Founded in 1989, Bloor Research is one of the world's leading IT research, analysis and consultancy organisations—distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services and consultancy projects.

About the author

Philip Howard
Research Director - Data

Philip started in the computer industry way back in 1973 and has variously worked as a systems analyst, programmer and salesperson, as well as in marketing and product management, for a variety of companies including GEC Marconi, GPT, Philips Data Systems, Raytheon and NCR.

After a quarter of a century of not being his own boss Philip set up what is now P3ST (Wordsmiths) Ltd in 1992 and his first client was Bloor Research (then ButlerBloor), with Philip working for the company as an associate analyst. His relationship with Bloor Research has continued since that time and he is now Research Director. His practice area encompasses anything to do with data and content and he has five further analysts working with him in this area. While maintaining an overview of the whole space Philip himself specialises in databases, data management, data integration, data quality, data federation, master data management, data governance and data warehousing. He also has an interest in event stream/complex event processing.

In addition to the numerous reports Philip has written on behalf of Bloor Research, Philip also contributes regularly to www.IT-Director.com and www.IT-Analysis.com and was previously the editor of both "Application Development News" and "Operating System News" on behalf of Cambridge Market Intelligence (CMI). He has also contributed to various magazines and published a number of reports published by companies such as CMI and The Financial Times.

Away from work, Philip's primary leisure activities are canal boats, skiing, playing Bridge (at which he is a Life Master) and walking the dog.

Copyright & disclaimer