WHY DATA MIGRATION IS SO IMPORTANT

If you are reading this whitepaper then I'm going to assume you are in the process of planning a new application adoption to replace an older legacy platform. It could be a CRM, an ERP, an accounting platform... basically anything that contains data. Now you may be in consideration stage, maybe you've reached budgeting and planning, or you could be knee deep in deployment and data migration is threatening to sink your shiny new application adoption. Even if you've had a failed application implementation, we have produced this guide to help you navigate the waters ahead. And trust me, those waters are choppy!

First, let’s get into the right frame of mind with a few statistics. Did you know that according to Experian, due to issues with data migration, only 46% of all new database implementations are delivered on time? Or that an incredible 74% of projects go over budget?

In fact, according to Merkle Group Inc. a whopping 63% of all CRM platform adoptions fail at first attempt and this is fairly similar to the stats offered up by ERP Focus who state only 40% of ERP projects are successful first time.

This seemed rather high to me. So, I took a look at every data migration project we've worked on at Qbase over the past 5 years to see why we were appointed to work on the project. Incredibly, the figures from Merkle and ERP Focus are bang on the money. We were actually appointed to 63% of all our data migration projects due to a failed implementation and in only 27% of projects were we involved in the initial planning stage. Wow! Imagine how much time, effort and money
PART 1: PLAIN SAILING OR CAPSZIZED PROJECT. THE SIGNIFICANCE OF DATA MIGRATION.

has been wasted over the years due to poor planning and inadequate focus on data migration.

Getting the data migration right isn’t going to stop your solution being one of these statistics if your application has been poorly scoped, specified or designed. But here’s the thing. It actually may help identify if it has been poorly designed so you have a chance to do something about it. How can I know this? I did some more analysis on those 27% of projects we were involved in from day 1. In 70% of those projects, the data migration track actively identified problems with the main implementation design. In fact in 20% of those projects, the design issues highlighted were so severe that the System Integrators were sacked from the project and new suppliers appointed. And potentially the best statistic is this, when we had done a landscape analysis before the main solution was designed, 78% of projects were delivered within budget. But more on this later.

DATA MIGRATION: THE BENEFITS

It’s clear that the risk of implementation failure is a huge driver for taking data migration seriously for any new system adoption. But let’s be more optimistic and explore the positive reasons for investing in data migration.

Data is one of your biggest assets

“You can have data without information, but you cannot have information without data.” – Daniel Keys Moran, an American computer programmer and science fiction writer.

Accurate and complete data will maximise the value of any new enterprise application. It will enable your staff to be more productive, your supply chain to be more efficient, your customers to be more engaged, your financial planning to be more accurate. Basically, getting data right means results on your bottom line. In the US for example, Forbes reported that 5% of the national output has been enabled by data. It’s probably why companies such as Western Digital are advocating businesses hold a data value on their balance sheet.
PART 1: PLAIN SAILING OR CAPSIZED PROJECT. THE SIGNIFICANCE OF DATA MIGRATION.

Better data means better decisions

"If we have data, let’s look at data. If all we have are opinions, let’s go with mine." – Jim Barksdale, former Netscape CEO

When you get your data right, the decisions taken by your business will be better. They are grounded in evidence, not opinions or hunches. This will lead to you capitalising on more opportunities and enabling positive changes that will have a direct effect on business performance. At the RNLI, which is a large UK Charity dedicated to saving lives at sea, we saw that when they leveraged accurate data on their supporters they were able to increase the amount of donations from supporters by 53% in one campaign.

Accurate data improves efficiency

“Where there is data smoke, there is business fire.” – Thomas Redman

Any application where the data is accurate, timely and trusted will invariably lead to efficiency and productivity gains. Let me illustrate this with an example. At a home shopping retailer in the UK, we were appointed to rescue a failed ERP migration. When we reviewed the project, we discovered the project team had carried out a working time analysis of a typical sales order telephone call in the UAT system. In their legacy platform it would take 4 minutes and 20 seconds to process an average sale. The business case for the new platform said this could be achieved in 2 minutes 45 seconds. The reality, due to inaccurate data and a patched design, it was taking an astonishing 8 minutes! The result after the data migration was redone? 2 minutes 40 seconds.

Legislative Compliance

“Privacy protection is a key element of customer confidence and a pillar of sustainable digital business development.”
— Stéphane Nappo

I know it’s a dull subject, but it’s importance cannot be understated. More and more data classifications are being subject to international regulatory control. GDPR in Europe for example has redrawn the rules on managing personal data. It is now the law that your data on individuals should be accurate, maintained and made available to the individual for review. So making sure you accurately migrate data into your new platform will ensure you are meeting local regulations.
THE COMMON MISTAKES

As you have seen, the business case for a considered, planned and expertly executed data migration should write itself. So why then do so many projects get it so wrong? Let's take a look at some of these reasons.

The SI will “take care of it”

The likelihood is you are taking one of two approaches to your new application implementation. Either you are outsourcing to an SI (System Integrator or Solution Implementor), or you are tackling an in-house deployment. Looking once more at the migrations we've worked on in the recent past, by far the most common approach is to appoint an SI. 82% of projects we've been engaged on took this route.

When we look at the failed migrations, there is a clear trend. In all but one, the SI had been appointed to manage the migration. The problem is, SI’s as a rule aren’t data experts. They are platform experts. Although we are seeing a huge improvement in the data migration capability and expertise of certain SI’s, it still shocks us to see the techniques others employ. Generally they use an approach we call “see what sticks” where they will perform basic mapping from your old data model, attempt to load it to your new platform and whatever doesn’t load they deal with by exception. There is no measure of accuracy, nor improvement in quality.

So here is practical tip number 1. If you are thinking of appointing an SI to do the migration as well as the implementation, ask them to walk you through their methodology, to break down and justify their resource estimates and ask for references of past clients where the platform is live and specifically ask these references about the volume, accuracy and quality of the data in their new platform.

Unrealistic expectations

Carrying out a good data migration is time consuming. As a rule of thumb it’s likely to take roughly the same amount of time to do the migration as it does to do the platform development. I often see raised eyebrows when I make this statement in meetings, but as a veteran of a significant number of migrations now, this is what you should expect.

So don’t overpromise to your project sponsors or board. Give yourself enough time to include a data migration workstream in your project. The length of time you need will of course increase if you are consolidating a large number of legacy data stores as the complexity of the transformations will increase.
PART 1: PLAIN SAILING OR CAPSIZED PROJECT. THE SIGNIFICANCE OF DATA MIGRATION.

Poor scoping

No matter what text you read on the subject of reasons for failed projects, poor scoping will feature prominently in that list. Either the requirements were incomplete or misinterpreted, the planning failed or the specification was sketchy. Either way, it was probably because your initial analysis and stakeholder engagement was compromised.

When it comes to the data migration, a process called a “landscape analysis” is going to save your bacon. I’ll cover this later, but essentially you need to invest the time to understand and define the location, volume, content and accuracy of your legacy data and business processes. And don’t forget to engage the data owners. Without consulting them I guarantee you will hit rough waters further into your application implementation voyage.

Resistance to change

“Don’t let a good idea get in the way of a better one” – Nick Kelly, Director of Technical Solutions. Qbase.

Any new application will result in widespread change. There are many white papers and guides out there that can assist you with change management. But in the context of new application deployment and data migration, I’m just going to bring “process reproduction” to your attention.

In the last failed migration Qbase were brought in to rescue, the reason for the failure was the SI had allowed the client to dictate the processes the platform should undertake which were based largely on the legacy processes of the enterprise. Their rationale was they had invested heavily in optimising these processes and by keeping them it would limit change and result in a smoother transition to their new application. They couldn’t have been more wrong. What actually happened was the SI had to “force” the application to carry out the old processes. This was despite much slicker, easier processes being available out of the box. This subsequently compromised the data model and the SI was patching data and processes in the platform which impacted on performance and data accuracy.

So what should have happened? The client should have been willing to make changes to their enterprise to accommodate the new application. A good change management process would have recognised these processes were more efficient and delivered more business value. As it was, they were busily recreating their old legacy database in their brand new cloud application. So the business value and opportunity was lost and the data migration became so complex it was almost impossible to execute. Now the client has appointed a change management office and implementation, attempt number two, is well underway and on target with a brand
new SI and Qbase as a data migration specialist on-board.

And confession time. Let's talk about plumbers with leaky taps. This is exactly what happened at Qbase a few years ago. We had made a considerable investment in a new lead generation through to project billing process recommended to us by an external consultant. The problem was, our new CRM didn’t use this process. So we forced it to. Six months later, we happened to spot the out of the box process in our CRM platform in a finance review meeting. It turned out it was actually better than the one we had commissioned. And the change to that process took us less than a week, but over the past 2 years has saved us approximately 400 hours of unnecessary admin on the one we had originally commissioned.
DATA MIGRATION APPROACH

The evidence is abundant as to why you should take Data Migration seriously but let’s focus on the practicalities of how you can achieve a successful data migration for your new application.

At this juncture, I must encourage you to get your hands an excellent book by Data Migration Guru, Johnny Morris, called “Practical Data Migration V2”. If you want more in-depth information on how to conduct a migration. This is the go-to text for any IT or business professional engaged on one of these projects and you can find it on Amazon or in most e-book stores. Some of the ideas and tested processes we use today and I will be covering here originated from that book.

Now back to the approach. Data Migration as a practice is ultimately a methodical and logical process coupled with robust data governance. The methodology is designed to be manageable, measurable and meaningful. So you will only engage in activities that serve a purpose and it will lend itself to integration with the main project no matter if you are pursuing a waterfall style or agile approach to project management.

Our recommended approach is a simplification of the Johnny Morris model and is illustrated here:

- **DISCOVER**
  - Landscape Analysis
  - Stakeholder Engagement
  - Scoping
  - Planning & Governance
  - Commissioning

- **DESIGN**
  - Approach
  - Mapping
  - Gap Analysis
  - Development

- **TEST**
  - Code Testing
  - Process Testing
  - UAT Testing
  - Performance Tuning

- **DELIVER**
  - Go-live
  - Post Launch Support
  - Integration with other technology

“Data Migration as a practise is ultimately a methodical and logical process coupled with robust data governance.”
PART 2: PLOTTING YOUR DATA MIGRATION COURSE.

DISCOVER

All too often the discovery phase of large application adoption projects overlooks legacy data. Sure, there may be some basic data profiling carried out on the primary legacy database, but we often see little heed placed on the results to inform the subsequent platform purchasing decision or project design. To carry out a successful data migration, sufficient time should be given to understanding the existing landscape, interviewing data owners and stakeholders and creating governance mechanisms. In this section I attempt to provide you with some guidance of how to achieve this.

Landscape Analysis

A landscape analysis is the identification, cataloguing and measurement of all the data and data processes in your company or organisation. It provides a means to properly specify your new solution to accommodate existing data assets and core processes. And knowing the data you currently run your business upon means you can accurately plan how you will move existing data and processes into your new solution.

Our advice is you should commission a Landscape Analysis before any decisions are taken on your chosen solution. That means doing this work before you go out to tender.

Why? Well, as anyone who works in IT these days knows, applications and data are as much about business process as they are technology. If you haven’t defined and mapped your existing business processes and data first, how can you make an informed decision on a suitable solution or technology? You should prepare your Landscape Analysis as part of your headline project scoping and before you approach any SI’s.

Let me give you an example, again from the UK not-for-profit sector. Approximately 8 years ago Salesforce CRM started to appear as a potential alternative supporter management platform. Innovation from the existing providers was poor and UK charities were desperate to move to the cloud to better integrate their digital provision. So it seemed like a natural solution to bring in Salesforce. Many charities did this, but they made that jump without first assessing their business processes. There is a tax relief aid called “gift aid” in the UK where a charity can claim a proportion of a taxpayer’s income tax back on any donation. But Salesforce was a US led platform with no provision for this. Those early adopter charities ran into this problem mid-project and it derailed at least 3 major implementations I am aware of. Had sufficient business analysis and landscape analysis taken place, they may have chosen another platform, or at least ensured a solution was tried and tested on Salesforce before commissioning these projects. Incidentally, today this isn’t a problem and there are lots of solutions for the gift aid requirement on Salesforce, but back then it cost some...
charities significant amounts of money.

A landscape analysis by its very nature is a process by which you will identify and catalogue all data stores. Using interviews, questionnaires, process maps and IT audits, the first job is to identify all of these data stores and document them as part of a high level schema. Remember, you may have a number of un-registered data stores such as spreadsheets or databases held by departments or external suppliers. Consider holding a data amnesty to identify these.

At this stage you only need to know a basic understanding of the relationships between different data stores. You will get more detail later.

You now need to know what data exists in each data store. This is called profiling and it’s a relatively simple task given the right tools. The objective is to identify the format, quantity, age, range and general health of your data. It’s also an opportunity for you to identify what is core data, and what is superfluous, probably machine generated data, that you won’t need in any migration.

The next stage is lower level schema analysis of each data store. The aim is to create Entity Relationship Diagrams (ERDs) within each store. Again, there are tools that can help you reverse engineer ERDs from the core database tables.

You should now be in a position to understand the functions and processes inherent in your data stores. If you have access to data dictionaries, use them to identify the function of tables, entities and processes and start to categorise them into useful pots of data. For example, if you were migrating to a CRM you may have people, places, transactions, communications and so forth.

Your next job is to create a measure of data completeness and quality. Referring back to the profiling data and combining this with your new functional understanding, your role is to identify the expected content and identify where the gaps and problems are. For example, for data on a person, does the gender flag match the suffix or title? You should also look to identify any areas of duplication in the data. You should document your results and prepare them as a digestible report. This will become essential as part of your migration as you will need to fix any issues you have identified. At Qbase we call these DQRs (Data Quality Rules), and we’ll cover this in more detail in the governance section.

Finally you need to pull together all of the above into a high level ERD. This is where you can demonstrate actual relationships and processes between data stores. It will give you a measure on the totality of the data in question, and if you have the capability, you should perform inter-store deduplication to further refine these measures.
If you are planning on running the Landscape Analysis yourself, you will need people with a number of very specific skills. ETL skills are essential to pull all the data into a working environment for subsequent analysis. Data Architect skills are needed for the identification and understanding of ERDs. Analysts are needed for the low-level data and process analysis and mapping. Reporting and presentation skills are required for the end report. Plus you’ll need people with good interpersonal skills at every stage. A landscape analysis isn’t just about the data. It’s about the people who manage and maintain that data and the people who have a stake in the data. Working with these people is essential for an effective landscape analysis.

And don’t forget, this is essential viewing for any potential SI vendors or partners as part of your Request for Information process as it will help them to quantify and understand your requirement better and be able to provide a more accurate solution and estimation for the work.

By the way. If you are considering skipping this step, then maybe this stat will help you to decide. According to US ERP specialist Trajectory Inc. an astonishing 72% of projects are delivered on time and within budget when a landscape analysis is carried out during a project scoping phase. Compare that to the 60% of ERPs that fail first time!

Stakeholder Engagement

Stakeholder engagement will commence at the start of the Landscape Analysis. There are four steps to take under the stakeholder management “umbrella”.

Stakeholder Identification
Stakeholder Analysis
Stakeholder Communication
Stakeholder Engagement

Obviously, it starts by identifying all potential stakeholders across your Enterprise. But remember, stakeholders don’t need to be internal. In our experience there will be several suppliers, partners or maybe even customers who need to be briefed on your project.
Having identified this long list, do some work to identify their significance to the project. Try using a power and importance grid like this to score each stakeholder. That will help you with the next step which is to define a communication strategy and level of engagement.

**Scoping**

When we talk about scoping in the context of this “perfect” data migration, we don't mean identifying low level requirements or user stories. That’s a task for the main project. Rather, you are scoping the legacy data and processes that need to be accounted for in your end solution.
It should only be high level and easily digestible by any member of the project team. Consider it more like the “brief” for the data requirements in your final solution.

Planning & Governance

For this white paper, I’m not going to get into the intricacies of project planning. There are far too many options and nuances to cover here and the type of planning you undertake will largely depend on the main implementation project. However, what I will recommend is the data migration be its own workstream in what-ever planning technique you employ. Depending upon the size and scope of the Data Migration, definitely consider employing a dedicated project manager just to cover this workstream and ensure it has sufficient visibility at project board level. I think I’ve done enough to convince you now that it shouldn’t be an afterthought, so you need to ensure other project stakeholders don’t see it that way either.

What I would like to cover in this section is the data governance and introduce you to a technique called “DQRs” or Data Quality Rules to give them their full Sunday title. Starting with the landscape analysis and probably right up to the point where you migrate, you are going to encounter a number of issues, problems, rules, tolerances, transformations and quality checks that need to be addressed for data accuracy, formatting and completeness. Identifying, implementing and managing DQRs will enable you to do this. A DQR is a set of business rules that govern each piece of data. It could be formatting rules, transformation rules, tolerance rules, hygiene rules, match and merge rules... the list goes on.

Going back to my previous example, you may have a DQR whereby the suffix of a contact is checked against the gender. If for example, you had me in your data as Mr Rob Jones but you had my gender marked as female, then there is clearly an issue with the data. But how do you know which element is the truth? Is it “Mr” or is it “female”. To resolve the conflict you could create a rule. There are usually many ways to do this. In this example you could look to the first name to see if it is highly masculine or highly feminine. Or you could look at the original source and use trust ratings? Or maybe you look at the number of times the piece of information has been provided. So in this case, if you have seen “Mr” supplied 10 times, but “female” was only supplied once, then you would choose “Mr”. The exact rule is your decision and will be appropriate for the data type and specifics of your data model and data origination.

For guidance, when we use DQRs at Qbase there are some
techniques we always employ. The first is you should be able to define the scope of the problem. The second is whatever the solution is you should be able to write it out as written English that any stakeholder could read and understand. Finally references to DQRs should be annotated on your code in your migration controller. That way, if you ever change a DQR, you can find out where they are used in your code to carry out change impact assessments before making the change.

Let's take a closer look at the governance process around DQRs. A DQR solution consists of three elements:

- Process
- Decision Making
- Documentation

The DQR Process

The DQR process is a workflow that allows a question, issue or rule to be raised and provides an end-to-end mechanism for resolution and commitment to data policy. It can be illustrated in this workflow:
PART 2: PLOTTING YOUR DATA MIGRATION COURSE.

The process starts by identifying the DQR. Anyone in the project should have the ability to raise a DQR, but there should be a DQR Manager employed who will provide an initial triage to see if the DQR has already been raised, if it is only an update to an existing DQR or even if it is a DQR at all.

Once accepted, the DQR Manager, working with the relevant Analysts, will seek to quantify the scope of the DQR. For example, how many records does this effect, what areas does it impact upon, which departments does it affect...

Once scoped a decision can be made on its priority. At Qbase we classify DQRs as a Level 1, 2 or 3. Level 1s are the most serious in that they have many dependencies and it affects a large number of records. Level 3s are likely to affect far fewer records and could, if needed, be ignored to be fixed at a later date, or dropped from the future data model altogether.

Having prioritised them, you can then start to design the rule. Here we want to know what are we going to do with the data to create the fix. For the migration you will need to make a decision about where to apply the fix. For the vast majority of DQRs you will “fix in flight” which means you will transform the data in some way as part of your migration code. However, for many DQRs you may choose to ignore the issue, delete the data in the migration, fix it in the legacy database or maybe even fix the data in your new platform after migration.

Having implemented your DQR you will need to verify it worked. If it didn’t, the DQR needs to go back through the process until resolution is reached. Finally, you will need to deploy the DQR and manage its implementation through annotating the code where you apply the DQR. So, think about a numeric system to name each DQR.

As you’ve probably noticed, I’m trying to illustrate some of the advice provided with stories and experience of working on these projects in the past. At one client, a department who held a lot of weight in the organisation we were working for, identified a small set of DQRs they needed to be included in the migration. Due to this department’s particular standing in the organisation and the sensitivity of the data in question, the project board intervened in the DQR process and stated that these DQRs needed to be classified Level 1 and should be immediately prioritised. You may have noticed, this bypassed the quantify stage. As we looked to design a solution to resolve these DQRs, we realised it would be extremely complex and could take up to 10 days of development time. However, due to nothing but curiosity, our lead developer decided to interrogate the data in question and found it affected less than 100 records. Then he hand typed over an example record to make the fixes. After measuring that time he subsequently established the whole volume of corrections could be made after the migration in just over a day by any of the admin team at the client. This shows the power and importance of taking each step and
that you don’t have to fix everything in flight.

**DQR Decision Making**

For most projects, you should be able to appoint someone as your DQR Manager who is empowered to take decisions on business rules with the collaboration of internal subject matter experts where needed. But for larger projects where data is particularly complex and where there may be many departments with diverse processes, you may need to employ a DQR board.

A DQR board is a group of people who meet on a regular basis to take decisions on more complex or significant DQRs. The board should consist of the DQR Manager, the Project Manager, a senior Implementation Developer, a senior Migration Developer and a business Director. There should also be a list of “Occasional Board Members” who are subject matter experts who are rolled in or off the board as the migration starts to cover their area of business.

In both cases decisions must be taken about business rules. In our experience you will face anything between 200 to 2,000 DQRs. So getting your decision making process right is imperative to avoid delays elsewhere in the project.

**Documentation**

Each DQR needs to be documented. This includes the identification of the issue or rule, the scoping and analysis, the business rule decision and the written logic of how the rule works and where it should be applied. To do this we create a DQR Database.

The DQR Database is a tool that facilitates the process. It allows DQRs to be identified, categorised, investigated, assessed and escalated. In effect, it is very similar to a support
The ticket process employed by most IT and IS departments in the world.

The DQR database provides a workflow engine that maps the process and allows individual DQRs to progress through the various stages through to implementation. It also works as a project management aid in that it allows tasks to be raised and assigned to individuals, with progress measured and communicated to the DQR Manager.

The database should also provide the reporting element of the DQR process model. At Qbase we now use a DQR dashboard where we use speedometers to see the number of DQRs being raised and closed on a daily basis, as well as a pipeline for the workflow and burn-down rates for individuals tasked with DQR scoping, design and resolution. It works really well and focusses everyone on the task in hand. I would completely recommend you do this for your project.

Finally, once each DQR is resolved, it provides a database of data rules and policies that can be referenced in the future. At Qbase we usually use Jira for running DQR databases as it is extremely low cost and facilitates both the workflows and collaboration requirements for the database.

Finally, remember, DQRs documented in this way can become a permanent artefact for future use and can be applied at any time, on any data set or new platform adoption. They will create data governance that lasts beyond your application implementation project.

**Commissioning**

The last thing to do as part of the discovery phase is project commissioning. I’ve called this out as a distinct step as this is your opportunity to ask yourself some questions. Now you know what you know about your legacy data, plus you may have an initial corpus of DQRs to review, how should you commission your data migration exercise? You have a number of options, but it’s decision time:

**In-house** – If you’re confident you have the skills, expertise and resource to carry out the migration using internal staff. Choose this option. Remember Data Migrations are “all-encompassing” so you will have to free those staff up from their usual responsibilities. If you don’t, I can almost guarantee you’ll compromise your migration and you’ll be faced with delays and budget overruns.

**In-house with help** – A popular choice is to bring in contractors to work alongside your internal resource. Often these contractors come with relevant expertise and a ready to use tool kit of resources to kick start your migration. However, be careful. Contractors need closely managing. They work on day rates and it’s unlikely they will agree to...
a set price, “turn-key” project rate. Some of them can also be unreliable in the sense if they don’t like the way you are running things, or they aren’t finding the work rewarding, or maybe they get a better offer. They can just leave with very little notice. We’ve been brought into a lot of migrations at an advance stage because of this.

Outsource to an expert business – This option gives you the benefits of having experts on the ground, lots of resource and experience and you won’t have to worry about contractors leaving as the business you contract with will ensure succession planning and consistency. It’s also likely they will agree to a fixed price contract and as data migration experts they are going to bring staff, technology and process that you won’t get from contractors. The flip side though. It will be more expensive than the contractor option.

Outsource to your SI – The easiest option is to roll everything up as part of the contract award for your SI to take care of. If your SI has dedicated migration experts that are full time employees, you should get all the benefits of all the previous option but with closer working with the solution design and implementation team. However, SI’s as a rule, aren’t data experts and generally their focus is on the main system implementation, not the migration. But if you find an SI who’s references check out who have all the dedicated staff, technology stack and robust processes, you may find this is the easiest way to migrate. One thing to note with this option. Check that your SI isn’t outsourcing this function. If they are, they are probably marking up the price so you should consider doing your own outsourcing.
DESIGN

Once you have completed your discovery and only after you have selected your new technology platform and potential implementation partners, should you then start to look at the data migration design.

The design will clearly detail how you will carry out the migration. This all starts with a decision about your approach.

### Approach

Before you commence any low level design, you need to take some decisions about your preferred approach.

### Big bang Vs Phased Migration

Generally there are two extremes of how to carry out a migration. Either you choose a date and make a plan that says on that date you will extract all your data from your legacy systems, prevent all updates to those systems, and run your migration code to transform your legacy data and load it to your new platform. This is called a big bang and it means that as soon as the data is loaded, you make your new application available for the users. It's clean and simple, but...
it’s not always possible and it does have more of a risk of being delayed as EVERYTHING has to be complete before you press the big red “migrate” button.

The other option is a phased approach. In this strategy you migrate portions of your legacy data and business processes to your new application. You would have parallel running of your new application and your legacy ones. This can work beautifully well if you have distinct, exclusive alcoves of data and processes that don’t draw from any common objects or feature any data dependencies. You can carve up these data segments and move them, and the departments who are responsible for that data, over to your new platform. Then move onto the next department or segment until eventually your migration is complete and you can contemplate legacy decommissioning.

However, in our experience this virtually never happens. It’s rare to have distinct business functions with no overlap in your legacy data which makes a phased approach much more complex. You now have to consider some form of data synchronization or cut-over facility where common data and objects are kept in sync. This can get complex, exponentially so, as the number of shared fields increases. In many cases it can become prohibitively expensive as the design and development effort becomes enormous.

But once you’ve decided on your approach, you’re ready for the real design work!

Mapping

Whenever I first speak to people about data migration, their first question is frequently “ah, you mean mapping?” Well, yes. That’s definitely part of it. But to do it properly and effectively it’s not usually as simple as just pointing one field at another.

The whole reason the practice of data migration exists is generally your new application will be substantially different to your old one. If it isn’t then you should be asking yourself some questions about if you really needed a new one. Yes, there will be commonality. People and places for example are generally similar in both, but outside of that, the data models, table and field structures, pick lists, calculations, stored procedures… well pretty much everything will be different which means you have a large transformation exercise to undertake and for that, you need to plot a course which means you need a map!

I could write an entire white paper on data migration mapping, but I’ll keep it simple and stick to the main areas. When you create your mapping you are doing four things:
1. Showing the direction of travel for data
2. Detailing the transformation that needs to take place
3. Orchestrating the transformation.
4. Identifying the verification rules to reconcile the data

The direction of travel of data is showing where a piece of data originates from and where you plan on sending it in your new legacy platform. This is what most people consider as mapping. There are two ways to tackle this and you will need to do both, but there is definitely an order to do them if you want to be efficient!

Always start with “right to left” mapping. It’s called this as a typical Western process flow will start on the left and move to the right. As you can see below, data starts in the legacy platforms and is pushed via a migration controller (we’ll cover this soon), to the target system.

That’s a “left to right” flow. But “right to left” means starting with the target system and working back from that to find the source data. You should always try to start with this approach as decisions on data formatting and content will have been taken in the platform design which have likely simplified or split out legacy fields and tables. You just won’t be able to always see this when you start with a legacy field, especially when one field, containing different values, is being split and shared across a number of new fields in your new database.

When you carry out your right to left mapping, it’s not going to help if all you have are lots of arrows pointing from one field to others. You need to define what is going to happen to that data to make it fit for purpose. i.e. How will you transform the data. Luckily you’ve already got a vehicle to help you here, your DQR database. Not all your transformations will be set as DQRs, but the majority should be. If you don’t have a transformation in your DQRs, unless it’s a simple one-to-one mapping, then you should raise a new DQR at that point so there is a record of what transformations need to take place.

However, many of the transformations may need other transformations to have taken place first before they can proceed. So as part of your design you need to consider this as part of an orchestration step. For example, a contact needs to exist for it to be a campaign member or opportunity. During mapping design you should start to look at objects
and fields to identify the order in which they may need to be transformed as they are a dependency for a later transformation. And if this weren’t complicated enough, you may even find there are system generated values in your new platform that are dependencies for your transformations. At this point you need a very close relationship with your SI to architect how you will get this data as part of your migration controller process.

Finally, you also need to consider counting the data you’re planning to move. Is it all accounted for and how will you know? This should be part of your map. If you have data being moved from one place to a number of places in the new system, with potentially some data being dropped, ensure you add in some simple maths to your map that your developers can use to check their code when they come to write it.

So that was just right to left mapping, but now you need to carry out left to right mapping. But that’s going to need another technique called a Gap Analysis.

Gap Analysis

The gap analysis is a great tool as part of your migration arsenal. Its purpose is to review all the data that is being left behind by your right to left plan and it works by answering two questions;

What data should populate these unpopulated fields in our new platform?
And, What legacy data hasn’t been accounted for in the new platform?

Let’s start with the easy one, the empty fields. Having worked through your mapping, it’s probable that you will be faced with a number of fields in your new platform that don’t appear to have any legacy data available to populate them. That’s OK and it’s why you run the design phase like this. Your job is to raise all of these as DQRs. That way they will be triaged and the subject matter expert and SI will provide an overview of what data they expected would be in there and if it already exists, where it can be located and what transformation rules should be applied to it. In many instances this is probably a field you have only just created and only intend to gather the data for it after the application is launched. Others it just wouldn’t have been obvious where the data was and they can help you locate it.
The other half of gap analysis is to identify all the data in the legacy systems that now appears to have no home in the new platform. Be warned, this is where stress levels in the project may spiral. If your requirements definition wasn’t up to par, this could result in lots of requirements needing to be added to your platform design at the eleventh hour. But to be fair to the Business Analysts out there, it can be almost impossible to capture and cater for every requirement, no matter how thorough the BA program was. Just make sure you build in a phase for these corrections in your project plan.

In actual fact, the likelihood is there will be lots of redundant legacy data identified by the gap analysis that you genuinely don’t need anymore. Don’t think that just because it exists, it needs a home. In actual fact, unless you absolutely need it as part of one of your new processes, it will have a home. Just park it in your data warehouse or data lake so you can access it if you do ever need it again. It’s probably only ever going to be needed by your analysts or compliance teams in the future anyway. Just make sure they can get to it if needed.

In both cases above, the “gaps” can be solved with the DQR program and once you’ve gone through that process you should now be in a position to start the development work.

“The “gaps” can be solved with the DQR program and once you’ve gone through that process you should now be in a position to start the development work.”
DEVELOPMENT

At the risk of sounding conceit, if you’ve done a good job on the design, development should be a relatively trivial task. The design has given the developers everything they need to produce the code that will create the transformation. But you’ll notice I used the word “should”. These things are never that straightforward and as the developers work, they will constantly encounter unforeseen issues or nuances. So they must be plugged into the DQR program and the DQR program must be highly responsive during the development phase to remove developer blockers.

The Migration Controller

But the development isn’t just about creating transformation code. In actual fact they are producing an application called a Migration Controller. Let’s take a look at the constituent parts of a controller.

Extract

The first job of the controller is to extract all the data from the legacy sources. Generally at Qbase we try to detach the controller from the source systems and include a staging database to act as an intermediary. It’s likely your legacy data is pretty big and trying to perform a single extract of the entire thing will be needlessly time consuming. Instead, using a staging database, you can keep a local copy of the data next
to the migration controller and using differential updates, come D-day, you only have to grab the latest differential, not the full file. Your migration controller therefore only needs to collect data from staging and that is easier to program.

**Transform**

We’ve covered this in enough detail already. Well almost. As I’ve mentioned, the transformation code should cover off your mapping design and take into account your orchestration design and have self-referential code to reconcile data on the way through with error notifications for issues, but don’t forget about code annotation and version control. Code annotation first. I guarantee you will be regularly quizzed about different elements of the transformation so you need to be able to answer these questions quickly by referring to documentation and pointing people at the notes on each line of code. I know it’s tempting to skip this step when the pressure is on, but don’t. It will save you further down the line. I guarantee it. Even more crucial is version control. Particularly if you have a number of people working jointly on code. Employ a tool such as Github to manage code versions and it will also help you to avoid branching or running out of date code.

Speaking of version control. Time for another story. On one project we worked on we weren’t carrying out the “duplicate identification” in a large CRM migration. The client had a specialist and long standing supplier identifying duplicate contacts in a database which also referenced out to a house movers databases. During the final test load something just didn’t look right in the reconciliations reports we were producing. The headline counts for loading had jumped from the last test even though the test file hadn’t changed. As duplicate rules were thankfully being managed by our DQR process, we could see the number of duplicates should have risen and the headline counts in the end file fallen due to the last batch of rules that had been added. When we interrogated this we found the specialist contractor had ran an old version of the deduplication code and were blissfully unaware of this. We then found out they hadn’t been running version control on any of their code and had inadvertently overwritten new code with old code due to a branching issue. And you have probably already realized, they also had no backward reconciliation process to compare their output with previous results. So the message is clear. Make sure you employ version control!

**Load**

The final part of a migration controller is to load the data. The load needs close collaboration with your SI. It is likely that a number of processes, automated tasks and stored procedures need to be switched on and off as part of any load. You can
program this as part of the migration controller load scripts, but the code and the account needs the relevant permissions to do this. Again, another lesson from the past. For a cloud ERP project we were involved in many years ago, the SI hadn’t given us all the permissions we needed to turn off a range of stored procedures relating to pick list generation. When we ran our first test, thankfully only on a small 10,000 row file, we were surprised to find that 24 hours after kicking off the load the development sandbox still hadn’t loaded the data. When we interrogated the logs we found the problem and got the permissions added. By the way, had we tried this with a full volume file the load would have taken over 4 years!

Your migration controller is essentially an application that you are creating, but it often helps if you create it in a recognized ETL tool that can work at scale. Due to the collaboration capabilities, robustness, available connectors and scalability, we generally use Talend here at Qbase. I would wholeheartedly recommend this technology to anyone engaged in any large scale migration exercise as you can then reuse elements of your migration code as part of any permanent integrations you need to create into or out of your new application.

**TEST**

If I can encourage you to take one lesson from this white paper, it would be this. Do a proper job of testing. It is so important. Ensure you have devoted enough time for it in your plans and don’t consider testing to be contingency time you can take to make up for over-runs in design and development. Why? Because if you’ve had over-runs in design or development, you DEFINITELY need more time for testing as you will have probably rushed something during the previous phase trying to catch up.

At the start of your project, as part of the planning phase, build out a thorough test plan. Consider what it is you would like to test, why it needs to be tested, how you will perform the test and what resources you need to devote to it.

There will be three areas of testing you need to consider:

- **CODE TESTING**
- **PROCESS TESTING**
- **USER ACCEPTANCE TESTING**

Let’s start with code testing. This phase is largely driven by your developers and facilitated by your Business Analysts. The objective is to test that specific elements of code are
performing in the way you designed them to. It combines peer review with reconciliation and exceptions analysis. It largely works at a micro level within the code (sometimes referred to as unit testing) and is designed to test specific functions. Here is the code testing process we go through at Qbase.

The second area of testing involves process testing. Here your objective is to test the performance and accuracy of your migration controller and migration processes. You will be testing the full end-to-end process from data extraction through to load. It will also be a test of how your code knits together as a full application so you can test the orchestration and carry out performance tuning to optimise the code execution time.

As ever with processes there are steps to consider.

**Coverage Loads**

As early as you can you should produce what is known as a "coverage load". This is a subset of data (definitely no more than 10,000 rows) and it does not need to be accurate in terms of the transformation. The coverage load serves two purposes. The first is to test that you can physically load data into your target application using the application load utility or third party tool. The second purpose is to provide the
Application Developers with a corpus of data to use and test in their system development and testing activities.

Volume Loads

Having successfully completed a coverage load, and once the majority of the target application is developed, you should then run a volume load. This is where you are attempting to load as much as the final file as possible. Again, at this stage, the data does not need to be accurate, that is not the purpose. Instead you are looking to estimate load times as part of the go-live planning and test your assumptions around orchestration. Expect a lot of rejected rows, failures and exceptions when you run this for the first time. However, what this test will give you is a good idea of what you’ve miscalculated, what improvements you need to make and if your architecture needs any optimization.

A number of years ago when we were engaged on our first ever major migration, and by that I mean our first system that contained over 1 million contacts, we ran our volume load testing and ran our migration controller confident we’d done a great job. We’d lined our first volume test up to coincide with the first round of User Acceptance Testing and we were pretty pleased with ourselves as the multiple coverage loads we ran worked like a dream. As we hadn’t ever properly ran an end-to-end test that resulted in us loading data our optimism soon turned to panic when we tried the load at scale. The load would take 27 days by our estimate. We had 3 days. Gulp! It turned out the Application Vendor’s data loading utility was the culprit. It couldn’t cope with the increase in volumes and had no bulk load facility. Had we carried out volume testing throughout the development phase we would have spotted this months in advance. As it was, the whole project had to be pushed back by six weeks as we worked with the vendor on configuring a third party tool and developing a series of intermediary staging tables to create a new bulk load process. When we asked them why this wasn’t in place previously, we discovered that the largest instance they had ever migrated into their platform previously was under 100,000 contacts. They had no idea it was going to be a problem.

Reconciliation Testing

Reconciliation testing will have been performed throughout the code testing phase, but it is always useful to run a macro view of reconciliation. At Qbase, we run volume and reconciliation testing on a weekly basis throughout development as it helps us optimise as we work. However, what you are particularly interested in here are rejected rows. This is data that didn’t make it through your DQRs or couldn’t be loaded into the target application. If you have used a blue chip ETL tool such as Talend, rejected rows reports should
come with a reason for rejection. These are gold dust for your DQR program and to save time in diagnostics.

And remember the golden rule of reconciliation. Things can be different, just as long as you can explain why they are different and that you are happy and confident in that application. If you’ve done a good job on your DQRs, this shouldn’t be an issue.

**Performance Tuning**

One of the areas you should be actively measuring when you run your coverage and volume tests is timing. You need to know how long the migration controller takes to execute and the velocity of the load. As part of the project planning you should have identified a “migration window” where you are able to execute the migration without bumping into critical business processes. Your objective is to ensure you optimise your code and the process so that you can comfortably run your migration within that window.

Performance tuning gives you that ability. You should be aiming for the migration controller to run in the shortest possible time and the load to run as quickly as possible. Your developers will help with the controller, but you will need the assistance of your Application Developers and product vendor to maximise the load velocity. Pay particular attention to stored procedures and processes in your application and take as much advice as you can from the vendor on what is possible and what they have seen done before to optimise load speeds. And if you are using a third party ETL tool, ensure it is one that is certified by the vendor and that it has off the shelf connectors configured to the latest version of the vendor application. Finally, this is lesson we learnt the hard way, if you are using dev environments or sandboxes, ensure these are as close to live conditions as possible. When we’ve been performance testing in many popular cloud based applications for example, we’ve found their development sandboxes are ran under constrained resources so you don’t get a true representation of performance. If you notify the vendor of your performance tests in advance, they are sometimes able to commission more representative environments for you to use for testing.

One other trick to remember when you are performance testing. You don’t necessarily need to run the whole end-to-end process every time. It’s quite common for us to create a milestone file with cut-overs or differentials that run to update the data that had already been successfully loaded. This way you can run the main bulk of the migration in advance of go-live and go-live then only needs to take care of the difference from the point where you last extracted the legacy data. It does need more development, but for migrations where billions of rows and millions of contacts are present, this is the only realistic method to carry out a
migration over a sensible migration window.

End-to-End Testing

By now, end-to-end testing should be a breeze for you. If you have been performing regular volume and reconciliation tests, plus performance tuning as you go, then you shouldn’t have too many surprises. However, it is still extremely sensible to recreate the planned “go-live” scenario to discover if things do run as smoothly as you anticipate.

My advice is to build in end-to-end tests to coincide with User Acceptance Testing. Working back from the date where UAT will take place, recreate the migration process and run everything as if you were performing the real migration. In my experience this is unlikely to wash out much that you didn’t know already from a development stand point, but it is incredibly valuable for communications and process testing. I remember running this once for an organisation where we were at our desks until late in the evening waiting for an update to be delivered from a legacy data store managed by the client. We were supposed to receive a notification when the file had transferred but we didn’t get one. It turned out the job had failed at the client’s end but they had left work after running the job and then forgotten to check the job had completed when they arrived home. They hadn’t created automated error notifications as they thought they had, so were genuinely surprised when they got an SMS from us at 9pm asking where the data was at which point he realised he also hadn’t had a delivery notification. This illustrates the power of testing the end-to-end process and how the smallest thing has the power to derail months, if not years, of work.

User Acceptance Testing

User Acceptance Testing or UAT will have been clearly built into your plans but it is important that you include data testing into UAT and brief your users on this.

Your users are the people that interact and use your data on a daily basis. They will have specialist knowledge in different areas of your enterprise and know what to expect when it comes to the data. It’s a good idea to prepare users before they arrive for testing by having them build scenarios that recreate what they are likely to be doing in the new application. As part of those scenarios, ask them to check the data and counts are within the expected ranges and that it looks accurate.

In our experience, as a very rough rule of thumb, approximately 35 – 40% of all “bugs” identified by users during UAT will relate to data. And of these approximately 25% will be genuine defects, the rest will be educational
exercises on your DQRs or change requests relating to DQRs.

As part of your issue and bug management processes, ensure you have a workflow to manage data bugs and ensure the Migration Developers and Project Manager are properly plugged into this. Here’s what our defect management workflow process looks like:
Go Live

Congratulations on making it this far. You’re now in the home stretch and believe it or not this is the easy part. Because you’ve been testing throughout and you’ve ran end-to-end rehearsals you know what to expect. But yep. It is still daunting and can cause a few butterflies in the stomach.

At this stage there isn’t much to tell you. You’ve planned for this and you know what you need to do. My biggest piece of advice is to engineer in monitoring tools and dashboards so you (and the many stakeholders) can keep abreast of the process. Build in milestones that turn green as they are achieved and ensure you’ve got notifications fully configured.

The other bit of advice is don’t go into go-live without a contingency plan. In actual fact, this is likely to be fairly simple. Just ensure you are able to fall-back onto your old infrastructure should anything go wrong. Well, that sounds simple but I do remember one of the first migrations I encountered in my career there was no contingency. I have no idea how this occurred as I wasn’t in IT at this point, but the business I worked for planned to introduce a new Oracle Back Office customer database over a bank-holiday weekend. When we arrived for work on Tuesday morning there was a very stressed, tired looking IT Team ready to greet everyone on reception to inform us we had no database. And I was asked to urgently create manual forms and spreadsheets to book business that day. This continued for 8 working days before our old platform was live again. Sales over that period were down almost 40% as the sales teams were hamstrung without access to the customer database.

“Engineer in monitoring tools and dashboards, don’t go into go-live without a contingency plan.”
Post Go-Live Support

There is nothing like releasing a creation into the wild to truly uncover the defects and problems. It’s why you need to ensure you have post go-live support for the data migration, not just the new application.

The best way to offer effective support in this area is to book at least one of your Migration Developers out after the point you launch. You need someone who intrinsically understands the DQRs and migration controller who has been exposed to the totality of your legacy data. When we plan our largest migrations at Qbase, we generally assign a single Developer on a full time basis for a month after the migration, then that drops to 50% time for month two and 25% for month three. After this point it will be highly unlikely you’ll receive any more migration support requests, but if you do, you can deal with them by exception and just use your support ticket helpdesk function to manage expectations on how quickly you can resolve issues. Which reminds me. Do ensure your Developer is plugged into your helpdesk and all the appropriate workflows have been created so relevant tickets can be assigned to them.

Technology Integration

I had considered including integration within each section of this white paper, but it would confuse the main narrative as it is in itself a distinct workstream that needs to be addressed. However, it’s also so important it couldn’t be ignored so I thought it a good idea to provide a quick summary of this topic, a mop up if you will, so that you can consider it as part of your wider application adoption solution.

On a surprising number of occasions, particularly where we have been invited in to rescue failed migrations, we’ve found integration of the new application with other technology and legacy data feeds was just expected to “happen”. People are often so focussed on the success of their specific application, they forget about the entire solution. It’s highly probable your application is a significant focal point for a wider operation and series of business processes. And with that comes a need to integrate your application with other applications, technologies, partners, feeds, services... the list goes on.

Ideally, your efforts around integration should begin during the full solution planning stage. I may even write a similar white paper to this one at some point about planning and implementing tech integration as part of your new database application adoption, and if I do, that’s where I’ll start. But today, I just want to give you a steer on things to consider.

“IT’s highly probable your application is a significant focal point for a wider operation and series of business processes. And with that comes a need to integrate your application with other applications, technologies, partners, feeds, services... the list goes on.”
When you are considering integration, particularly if you are adopting a cloud first application, it is likely there are hundreds, if not thousands, of off the shelf integrations to get it talking to other applications. However, you are also likely to find there are many essential integrations without an off-the shelf solution. The good news, and your main takeaway from this section, is this; you can probably use the ETL tool you built your migration controller in, along with much of the transformation and load code, to carry out these integration tasks. Make sure you identify these as part of your landscape analysis as it will help scope the right solution for you for that requirement and your data migration.
FINALLY, SOME SAGE ADVICE

I mentioned the fabulous Johnny Morris book on Data Migrations earlier (Again, do look into this if you have found this white paper useful), and a lot of the process I've discussed here has grown from his teachings so I thought it would be useful to introduce you to his golden rules for a successful data migration.

Data migration success factors

- **FOUR GOLDEN RULES**
  - Data migration is a business issue
  - The business knows best
  - No-one needs perfect data
  - If you can’t count it, it doesn’t count

- Flexible, business-driven migration solution
- Successful Data Migration
- Experienced Staff
- Standardised methodology
Rule One: Data migration is a business not a technical issue

I’ve mentioned this a few times already, and many of the techniques I have introduced have been designed to capture participation from the business, particularly the DQR process and UAT testing. But to help with this when you are considering the project scope, no matter what or where the legacy data currently resides, ask yourself and your colleagues these three questions:

• Why are we migrating this data?
• What data should be migrated?
• When should it be migrated?

You’ll notice these are questions for Business Managers, not Technicians, but without answers to them the Technicians and Developers can’t do their job!

Rule Two: the business knows best

The second rule of successful data migration is closely linked to the first. This is that the business drivers, not technical ones, should take precedence. It is critically important that business goals should define the solution and approach selected, and not the other way around. Therefore, to be successful, the chief business stakeholders must define their requirements and take responsibility for driving the project.

But, and this is a big but, if you are buying an off-the-shelf application, before you ask the business to drive the approach, have a look at the processes and solutions in the application first and brief your business on them. My advice is, you should where-ever possible try to use these first and only if they are definitely not suitable should you then bespoke any element of your application to meet the business requirement.

Rule Three: no one needs, wants or will pay for perfect data

Applications are only as good as the data they have available to them. We also know that many a data migration has been scuppered by overestimating the quality of, or not understanding, source data. Oh the joy of legacy data with its gaps, inconsistencies and redundancies! However, while enhancing data quality is a worthy goal, it is really important not to go off on a tangent mid-project in the quest for perfect data quality. Like over-specification of an application, the quest for data perfection can result in negative consequences for the project. It is where many, many projects run aground – inflating both the cost and the time to deliver the project.

Rule Four: If you can’t count it, it doesn’t count

Another challenge is how to measure data quality in order to assess the state of your legacy data and determine the level of quality your business users require. To make matters
worse, data quality is not static but erodes and improves over time. It is really important that the measures you use make sense to business users and not just to technologists. This enables you to measure deliverables, perform gap analyses, and monitor and improve ongoing data quality. It also ensures that you are concentrating your efforts on where business users see value and can quantify the benefits.

IN CONCLUSION

Introducing any new database application solution at scale will be a massive undertaking. It offers tremendous opportunities for your enterprise but carries with it huge risk. It is imperative that you have a considered approach that pulls on tried and tested methodologies across all elements of your programme and this includes Data Migration. By following a standardised methodology or employing an experienced data migration partner such as Qbase, you are moving the odds in your favour to have a successful deployment. To reiterate, you move from a 1 in 3 likelihood of success to a 4 in 5 probability!

Good luck with your new solution and if you need any help, then please get in touch. You can e-mail me at rob.jones@qbase.net or connect with me on LinkedIn. Just search for qbasesrob.

ABOUT QBASE

At Qbase we are data experts. We help our clients migrate, integrate, manage and deploy data to deliver innovative, robust and effective solutions. If you need advice or assistance with a data migration project for your organisation then we would love to hear from you. Everything we have covered in this white paper are services Qbase offers and we can help you whether you are just scoping a new solution or need us to rescue a failed deployment.

ABOUT THE AUTHOR

In his 25-year career, Rob has headed up business functions for a number of UK companies and organisations. Since joining Qbase eight years ago, Rob has provided strategic guidance to many of the UK’s largest enterprises. Rob specialises in using technology, data and analytics to power innovative business processes. He is best known for his expertise in leveraging the latest marketing technologies but he has worked on some of the largest CRM, ERP and Marketing Database projects in Europe.