TDWI

MONOGRAPH SERIES

APRIL 2006

Best Practices in Data Migration

By Philip Russom

Senior Manager of Research and Services
The Data Warehousing Institute

tdwi
THE DATA WAREHOUSING INSTITUTE

Best Practices in Data Migration

Table of Contents

About the Author

PHILIP RUSSOM is the senior manager of research and services at The Data Warehousing Institute (TDWI), where he oversees many of TDWI's research-oriented publications, services, awards, and events. Before joining TDWI in 2005, Russom was an industry analyst covering BI at Forrester Research, Giga Information Group, and Hurwitz Group. He also ran his own business as an independent industry analyst and BI consultant, and was contributing editor with *Intelligent Enterprise* and *DM Review* magazines. Before that, Russom worked in technical and marketing positions for various database vendors. You can reach him at prussom@tdwi.org.

About Our Sponsor

INFORMATICA
The Data Integration Company™

Informatica Corporation delivers data integration software and services to solve a problem facing most large organizations: the fragmentation of data across disparate systems. Informatica helps organizations gain greater business value from their information assets by integrating their enterprise data. Informatica's open, platform-neutral software reduces costs, speeds time to results, and scales to handle data integration projects of any size or complexity. With a proven 13-year track record of success, Informatica helps companies and government organizations of all sizes realize the full business potential of their enterprise data. That's why Informatica is known as the data integration company. For more information, call 650.385.5000 (800.653.3871 in the U.S.), or visit www.informatica.com.

Executive Summary

Data migration is a prominent data-movement technique that's commonly combined with other techniques.

Many IT projects today concern some kind of corrective measure at the platform level, whether integrating application silos, upgrading packaged applications, consolidating redundant IT systems, or migrating data and applications from old to modern platforms. These project types are related because they all involve moving data from database to database or application to application. Furthermore, they all require similar data modeling and data integration skills, and users apply one or more project types together. Among these, data migration is prominent because it is both a discrete project and a common component of other data-movement projects.

Data migration is rarely a one-way trip from point A to point B.

Data migration suffers from a few myths and misconceptions. For example, it rarely copies data once, one way, from one system to another. Instead, a project of any complexity transforms data substantially and involves multiple data sources and targets. Data may exist in old and new systems simultaneously, requiring that the two systems be synchronized for months or years.

Successful migrations include data profiling and data quality.

In many ways, data migration is a specialized form of data integration, so it relies heavily on integration technologies and practices. Like all data integration practices, data migration benefits greatly from data profiling, both in early phases (to discover and analyze source data) and later ones (to verify that data migrated correctly). And, of course, data migration inevitably ferrets out data quality issues, which can be exacerbated by the legacy nature of the data involved, so every data migration project must include software automation for correcting and enhancing data.

Best practices for data migration must support its iterative nature.

Many of the best practices of data migration center on its iterative nature. Development is an iterative cycle of profiling, analysis, design, and testing. Development best practice issues involve profiling arcane legacy data, modeling targets, developing transformations, and creating test datasets. Deployment is iterative, too, involving multiple rollout and handoff phases that may span years. Its best practices involve coordinating the migration's iterative rollout with end users, synchronizing old and new systems, handing off the new platform to its new administrators, and retiring the legacy platform. Some of these are business issues, and may require collaboration and consensus between IT's migration team and the line-of-business managers affected by the migrated systems.

ETL is the preferred technology for data migration.

Combining all of these issues makes data migration a complex and demanding project that's best accomplished using a feature-rich data integration tool that supports iterative methodologies. Although several technologies are available for data migration and similar projects, extract, transform, and load (ETL) is by far the preferred one. In a recent TDWI Technology Survey, 41% of respondents chose ETL over hand-coding (27%), replication (11%), and EAI (3.5%). Users prefer ETL for its unique ability to handle the extreme requirements of data migration, including terabyte-scale datasets, multi-pass data transformations, deep data profiling, interoperability with data quality tools, and many-to-many data integration capabilities.

Defining Data Migration

Countless executives complain that they're drowning in data, frustrated because they can't realize the value of this resource. But what feels like drowning is more like buckets of rain falling from every isolated application, database, portal, and spreadsheet. And the buckets won't fit in the same rain barrel, because each has a unique data model, age, relevance, and quality. Trying to manage a business efficiently or make an enlightened decision based on dozens of data sources is like trying to bathe in a dozen buckets of varying depths and cleanliness.

For many user organizations, this decade has been about addressing problems of this sort while chanting "do more with less." On the one hand, this mantra tells us to wring more value from preexisting IT systems before considering new ones. On the other hand, it leads us to lessen the number of IT systems, whether to reduce costs or to focus information for greater visibility.

Regardless of which direction the "do more with less" mantra leads, it usually manifests itself in IT projects for the migration, consolidation, upgrade, and integration of databases and applications. These project types are related because they all involve moving data from database to database or application to application, and they require similar data modeling and data integration skills. They are also related because users commonly apply one or more project types together.

Despite their similarities, data migration, consolidation, upgrade, and integration differ in two areas:

- **Number of data sources and targets.** The flow of data from its sources to its targets can be described as many-to-one (consolidation), one-to-one (migration or upgrade), and many-to-many (integration). This enables us to differentiate these similar data-movement techniques.
- **Diversity of source and target data models.** Much of the time spent in data-movement development concerns mappings and transforms between source and target data models. The more divergent they are (or the greater the number of data models involved), the more time development will take (see Figure 1).

USER STORY
Large projects may require multiple data-movement techniques executed in multiple phases.

Early this decade, a manufacturer decided to consolidate the 17 instances of ERP applications that it had collected through mergers, acquisitions, and the independent IT budgets of its subsidiaries. They first migrated non-standard ERP applications to the ERP standard they had chosen, then upgraded all instances to the same version. Eventually they consolidated these into three instances, one each for the Americas, EMEA, and Asia-Pacific, with application integration and data integration synchronizing the three.

In this case, data migrations and application upgrades were preparatory steps for the consolidation of ERP applications and their data, with integration connecting instances that could not be consolidated further. Note that achieving application consolidation required considerable work in migrating, consolidating, upgrading, and integrating application data.

Migration is the most pervasive data-movement technique.

Regardless of the type of data-movement technique selected for a project, data migration is commonly required. For instance, data consolidation is rarely as easy as shuffling two decks of cards together; one deck must migrate to look like the other before consolidation can occur. Ideally, an application or database upgrade should be a simple copy operation, but it's more like a migration when users have customized the system or when versions differ significantly. Even data integration—which is about capturing changed data incrementally—starts with a first-phase data dump or data initialization process that resembles a data migration.

FIGURE 1. Related types of data-movement solutions.

Type of technique	Number of data sources and targets	Diversity of data models	Examples of data-movement techniques
Consolidation	Many to one	May be homogeneous or heterogeneous	Consolidating multiple instances of a packaged application into one involves many homogeneous databases, unless they've been customized. Data mart consolidation is homogeneous in that they're all marts, yet heterogeneous in that each mart has a unique data model.[1]
Migration	One to one	Always heterogeneous	Migrations typically abandon an old platform in favor of a new one, as when migrating from a legacy hierarchical database to a modern relational one. Since platforms differ sharply, data models will differ.
Upgrade	One to one	Usually homogeneous	Upgrading a packaged application for ERP or CRM can be complex when users are two or more versions behind or have customized the application. This kind of upgrade is more like a migration. Others are simple and homogeneous.
Integration	Many to many (or any subset)	Extremely heterogeneous	Compared to other data-movement solutions, integration is noninvasive (doesn't destroy a platform) and quick (doesn't require as much target development). But it's challenging when sources and targets are numerous and extremely heterogeneous. Note that data consolidations, migrations, and upgrades are commonly executed with data integration technologies like ETL and replication.

Migration Myths and Misconceptions

Given its complexity and similarity to related techniques, data migration suffers a few myths and misconceptions that merit correction here:

Data migration is not a matter of copying data.

The term *migration* misleads some folks to think they can simply copy data from system A to system B. But the systems usually have different data models, so mapping and copying data without transforming it is rare. To the contrary, almost all migrations involve extensive transformations of data, simply to get the data from one model to another. Savvy IT organizations consider migration an opportunity to improve data models and data quality, although this requires even more transformation. In an application migration, data representing the business process automated by the old application must be radically repurposed to fit the automated processes of the new one, thus requiring extreme data transformations.

Data migration is not a one-shot task.

In the context of a single data migration project, it's common to have multiple project phases. This is a good engineering practice that breaks the migration of data and the rollout of the new application to end users into manageable chunks, instead of one risky "big bang." With multiple phases, it's possible that the same database may be migrated and synchronized with a target database multiple times. In some cases, the first attempt at data migration fails, demanding another shot.

Data migration is not necessarily a one-way street.

It's likely that an old application will remain in use while its new replacement is also in use. This happens when an early phase of the rollout delivers the new application to a limited end user population, and a different user population continues to use the old one until they are migrated. With two redundant applications in production simultaneously, synchronizing data between them is a required, two-way data operation. When the handoff period is long, this is a multi-year task.

[1] In this paper, the term *consolidation* refers to a process where diverse data models are merged into a single data model in one database management system (DBMS) instance. *Collocation* simply copies multiple databases into a single DBMS instance (typically on a single hardware server) without merging multiple data models. Collocation helps reduce IT administrative costs, but it doesn't solve the information silo problem like consolidation does.

Data migration is not a one-time task.

Data migration is something that companies of any size and age do repeatedly. Organizations that have numerous legacy and redundant systems can easily spend years migrating and consolidating them. Some companies put off a needed migration for years, waiting for technical resources to free up or for a legacy platform to depreciate. Even if an organization has polished its enterprise data architecture via upgrades, consolidations, and migrations, a couple of acquisitions will make them do it all over again.

Data migration merits a permanent investment, not a temporary reassignment of resources.

Assigning dedicated resources to data migration is recommended in organizations that face legacy, siloed, and redundant systems on the technical side, or mergers, acquisitions, and partnerships involving data exchange on the business side. Since data migration shares requirements for technologies and personnel skills with data integration, consolidation, quality, system upgrades, and enterprise data architecture projects, the investment should gain efficiency by addressing all these in a common organizational structure. That structure could be tightly focused, like an integration competency center. Or it could assume a broader mandate, like a data governance committee. Regardless of which form an organization's investment takes, it should collect and support the personnel and tools—plus the budget and executive mandate—required for successful data migration and similar work.

USER STORY
Acquisitions weaken data management functions without a dedicated migration and consolidation team.

A U.S. retail bank experienced organic growth for years by developing and retaining its customer base. But organic growth wasn't fast enough to keep pace with competition, so a few years ago the bank's management began acquiring smaller competitors. At first, the bank's data warehousing team was glad to handle the ensuing data migrations (which are key to a successful acquisition), because their involvement allowed them to greatly improve the data, thus protecting the data warehouse's quality and accuracy. But a few acquisitions turned into a business model for growth, and now a continuous stream of acquisitions is likely. Data warehouse and business intelligence development is sitting on the back burner, leaving users who depend on them in the lurch. The data warehouse director is now lobbying for a dedicated data migration and consolidation team.

Selecting Technologies for Data Migration

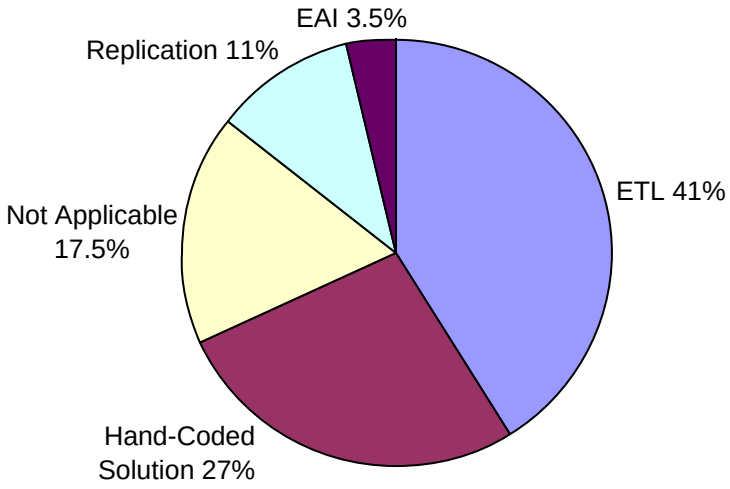
Multiple technologies and best practices are available for use with data migration, and more than one of these may be useful to a single project. Every data migration will have a unique combination of preexisting systems, plus requirements for the new system and its end users. Then there are the technology requirements common to all data migration projects (described next). IT should select technologies based on these factors, tempered by a consideration of how often and deeply they will perform data migration and similar projects in the future.

Several data integration tools and practices apply to data migration.

Extract, transform, load

- **Extract, transform, and load (ETL) is the preferred technology for data migration.** In a recent Technology Survey, TDWI asked: "When your organization consolidates or migrates databases, what is the preferred technology for most projects?" For 41% of respondents, ETL was the preferred technology, ahead of hand-coded solutions (27%), replication (11%), and EAI (3.5%). (See Figure 2.) Users prefer ETL for its unique ability to handle the extreme requirements of data migration, including terabyte-scale datasets, multi-pass data transformations, deep data profiling, interoperability with data quality tools, and many-to-many data integration capabilities.

FIGURE 2. Users prefer ETL over other technologies when migrating data.[2]



Hand coding

- **Hand coding appeals to some, despite a lack of productivity.** Studies have shown that tool-based data integration development and maintenance is far more productive (and, therefore, more economical) than hand-coded solutions.[3] Yet, hand coding persists because developers can't wean themselves of it, consultants use it as an excuse to rack up billable hours, and short-sighted managers won't spend in the near term to get the long-term cost reductions of tool productivity. It's time for everyone to do the math and recognize the economic superiority of tool use—at least for data integration projects.

Replication

- **Database replication is easy and accessible, but misses some requirements.** All data management professionals are conversant in replication, and a fair amount of replication functionality comes at no additional charge with a relational database license. This kind of low-end replication is usually limited to moving data one way without transformation between instances of the same database brand. Some high-end replication tools (bought separately) provide bidirectional, transformational, heterogeneous data synchronization, which is required when old and new databases of different types operate concurrently.

Enterprise application integration

- **Enterprise application integration (EAI) is not appropriate to data migration.** EAI excels at very quickly moving small amounts of information between the logic layers of applications. But EAI tools cannot handle the extreme volume, transformation, profiling, data quality, and many-to-one integration requirements of data migrations.

Data profiling and data quality go hand-in-hand with data migration.

Data migrations, like other uses of data integration technologies, benefit from data profiling up front and data quality measures applied in deployment:

Data profiling

- **Profile deeply up front or suffer unpredictable setbacks later.** It's hard to rationalize the time and resources committed to data profiling, since this is not the actual deliverable. But avoid the urge to scrimp on data profiling, because its benefits include more accurate scope,

[2] Source: TDWI Technology Survey, February 2006. 166 respondents.

[3] "For the wide majority of cases, developing a solution atop a vendor's data integration tool achieves monetary and time savings over hand coding." According to the Forrester Research report *The Total Economic Impact of Deploying Informatica PowerCenter*, January 2004.

better defined business benefits, and a reduction of "gotchas" that pop up in testing and deployment.

- **Rely on profiling for assuring that migration succeeded.** A key but underutilized use of profiling is to audit and reconcile migrated data. In other words, profiling is not just for source data, but also for newly migrated data. Profiling can compare source and target applications or databases to validate that the data was migrated properly.

Profiling tools

- **Use profiling tools instead of manual methods.** Users profile data via manual methods, dedicated profiling tools, or the profiling functions found in most modern data integration and quality tools. Manual methods are inferior because of the time-consuming and error-prone process of moving profile information from a query to the documentation to the data quality solution. When possible, users should profile with a vendor tool to get greater accuracy, repeatability, and productivity.

Data quality

- **Improve the quality of data and metadata; don't just migrate them.** Like all data integration techniques, data migration exposes data quality issues, whether they are defects requiring correction or opportunities meriting leverage. The older the dataset and the more subject it is to data entry (the leading origin of defective data), the deeper its quality issues will be. Legacy data is prone to tough problems like multi-value fields, asymmetric hierarchical structures, orphaned data structures, and multiple data standards (which evolved as the application or database was revised).

  Hence, many integration technologies performing a data migration will call out to data quality tools that perform functions like name-and-address cleansing, deduplication, and standardization. With ERP migrations and consolidations, product data can be improved by appending D-U-N-S numbers to supplier records and fuzzy-matching supplies that are equivalent though differently named.[4]

Redundant platforms create software and hardware issues for data migration.

Let's define *platform* as a combination of server hardware and server software. Formidable issues stem from the fact that both old and new platforms must run concurrently long enough to perform, test, and certify all phases of the migration. This redundancy increases the cost, complexity, and coordination of migration efforts:

Software issues

- **License renewals and extensions may be required for legacy and integration software.** The legacy platform must remain operational long enough to complete all migration phases and to accommodate a possible roll back in case of failure, so renew or sunset legacy software licenses accordingly. Also, a license may be required for the integration software to be used in migration, even if a license already exists for another purpose. For example, some ETL tool licenses are specifically purposed for data warehouse usage, and require a new license for work outside data warehousing.

Hardware issues

- **Redundancy and disposal are common hardware problems with data migration.** Most migrations involve a change of hardware platform. So-called legacy migrations may actually be driven by this change. Furthermore, almost all migrations require redundant hardware, so that both old and new platforms can operate concurrently for a while. Even more redundancy

[4] As Colin White points out, "Data quality issues are the leading inhibitor to successful data integration projects." See the TDWI report *Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise*, November 2005, available online at www.tdwi.org/research.

results when development and test environments recreate the old or new operational environments. Be sure the budget and deployment plans accommodate hardware redundancy. And get permission and a date for retiring old hardware, because this may be contingent on leasing, depreciation, and amortization schedules.

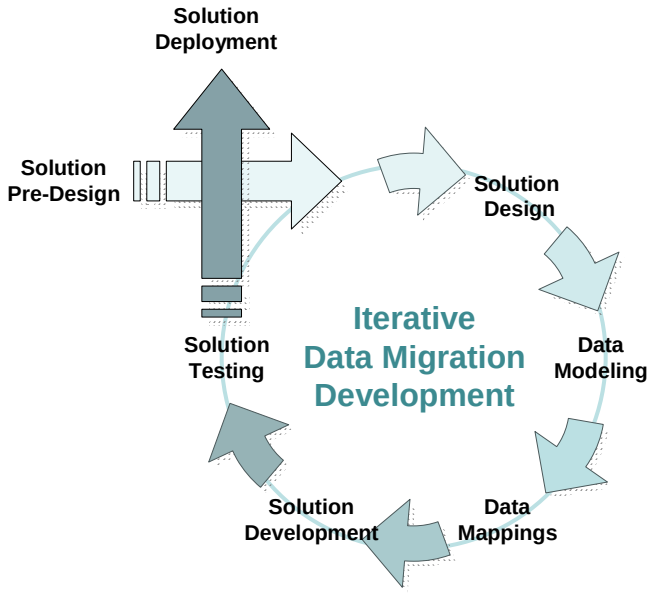
USER STORY
Application migration and consolidation are a common penance for the sins of the 1990s.

A midsize software vendor in New England went on a shopping spree in the late 1990s—as did many companies—and bought and built several applications for internal use. Most of these applications overlap, because they automate customer-oriented functions like order entry, shipping, billing, direct mail, customer base analysis, and multiple approaches to sales force automation. Today, this firm is paying for the indulgence of the 1990s by migrating and consolidating these redundant applications into a single instance of a single packaged application brand, plus extending that instance to fill in missing functions. This is a multi-year, multi-stage project that employs every technology mentioned here, as well as extensive process reengineering to ensure that end users and the business get full value from the new application.

Preparing to Migrate

Developing a data migration solution includes several steps that may be repeated iteratively. This is similar to the development cycles of most data integration projects, where some steps repeat—steps like architecting the solution design, modeling data targets, mapping data movement and transformation, developing a solution that pulls them all together, and testing. Pre-design activities like data profiling and requirements gathering precede the iterative cycle, whereas deployment follows it (see Figure 3).

FIGURE 3. Developing a data migration solution is a cyclical process.



Solution pre-design

The average data migration project is like deporting a nation of people, then nuking their country so they can never go back. Hence, the usual pre-design activities of requirements gathering and business coordination must be done with great sensitivity to assure business end users and their managers that this risky exodus will take them to a promised land. The best way to build their confidence is with a detailed project plan and timeline that call out those deliverables that they consider success factors.

Also, don't assume you intuitively understand the meaning and usage of all data you discover while profiling. Review data profiles with line-of-business managers and domain experts. They know how a field in an application graphical user interface (GUI) is really used, which can differ from its original design and how metadata describes it. Make this an opportunity for improving (or at least clarifying) the meaning and use of data.

Solution design

Some data migration projects execute as a single fell swoop. But most avoid such risky "big bangs" and follow good engineering practices by breaking the project into a series of manageable chunks. Dependencies may suggest which chunks should be handled separately and in what order. For example, some applications need to migrate to a standard platform or be upgraded before consolidation can begin. The challenge of solution design is to determine which tasks must be handled separately, as well as their optimal order and how to approach each one. Complex projects need a dedicated project manager who can write and enforce a project plan that documents the timeline for phases and their deliverables.

Data modeling

The amount and type of target data modeling varies greatly. At one extreme, migrating to a packaged application usually means repurposing legacy data to fit the predefined data model of that application, which is limiting, unless application customization is an option. At the other extreme, migrating to a newly designed, homegrown application means more data modeling work from scratch; yet, this allows more leeway in satisfying requirements like high performance and the unique set of processes that will access the database in your organization.

However, modeling the final database target is only part of the work; this task is complemented by the interim databases that need modeling. For instance, most data migration solutions follow a phased approach, so you may need to model data structures that roll out in phases. Other projects involve data from different sources arriving at different times and needing different types of processing—thus requiring models for data landing and staging databases, plus related structures like directories on network hard drives and FTP sites.

When possible, make data modeling an opportunity for improvement. After satisfying the requirements of applications that feed the database, satisfy those that read it. For instance, improve queries for data integration, data federation, operational reporting, and so on by enhancing the data model with materialized queries, database views, indices, look-up tables, and fields appended to records (say, calculated values that serve as performance metrics). Since metadata in legacy systems tends to include arcane acronyms and abbreviations—or generic descriptions of dubious meaning—making metadata as meaningful and business-friendly as possible will improve later tasks like reporting, administration, and application revisions or upgrades.

Data mappings

Even in straightforward data integration projects, data mapping is never as simple as connecting the dots. With data migration, relatively little data is merely mapped and copied. Most mappings include data transformations, because source and target data models are always different. Transformations of legacy data tend to be complex, because some source fields contain multiple values that must be teased out and mapped separately. Furthermore, legacy data will have missing values that can be filled in from other sources. When data landing and staging areas are involved, these increase the number of mappings. And then there are the usual transformations for standardization and quality, plus rules for conflict resolution, exceptions, and branching.

As a result, data mappings are always complex, consisting of many components that must be designed and tested individually before being strung together in a data flow. Given the complexity and the reiterative design method, users should avoid hand coding and do this work in the GUI of a dedicated data integration tool.

Solution development

Profiling a source database may reveal application logic in the database. This varies greatly, ranging from simple stored procedures for data verification or synchronization to full-blown procedural applications (especially in monolithic application architectures deployed before client-server architecture divided the monolith into tiers).

The question is, how much of this logic must be repurposed into the new platform, and where should it go? The trend (for 15 years now) is to move procedures out of the database tier and into the application tier—except procedures that are purely database operations. But this is a decision that database and application people must make together, because it affects the entire technology stack of the target application. Either way, converting database-specific, logical procedures is a large part of data migration development, so don't underestimate it.

Solution testing

In an ideal world, a data migration project will follow software engineering best practices, like isolating separate environments for development, test, and deployment. With data migration, however, the deployment environment is usually the target application. And development is inherently entwined with data profiling and iterative testing of mappings and transformations, so it often makes sense to consolidate data migration development and testing into one environment.

Early on, decide how much data from which sources to develop and test with. When they are modest in size and are useful off of their original platform, use complete source datasets. When source datasets are large, it's more efficient to work with a subset that is a small but representative sample of source data. If you choose to develop and test with a data sample, look for functions in vendor tools for integration, profiling, and quality that can sample a dataset to generate a test database. Note that HIPAA and other data privacy regulations may require you to make the test data anonymous. The same regulations may prohibit you from working directly with "live" data in the legacy platform, which is another reason to develop and test with an anonymous data sample.

Making Your Move

Once the data migration solution is developed, rolling it out concerns populating it with data, handing it to administrative personnel, bringing end users online, and monitoring its usage and load to ensure that performance, data quality, and user acceptance are adequate.

Deployment

Phased solution design is the preferred method for a data migration project, but the phases are not only about technology. This method also defines when each phased deliverable will be rolled out to which group of end users. Hence, the data migration designer must work with line-of-business managers to create a timeline that's reasonable for both IT and the business unit.

Administration

When the new platform goes into production with live data and active end users, the responsibility for its administration passes from the development department to another IT department (typically database administration or field IT). This is yet another organizational unit with which development must coordinate phased deliverables.

Synchronization

When both old and new platforms operate simultaneously—with live data and active end users—data integration processes must synchronize data across the two platforms. When batch processing during off-peak hours is appropriate, ETL and SQL-based replication are suitable technologies for data synchronization. If data must be synchronized in real time as it is created or changed, then the suitable technologies include EAI or transaction-based replication. Whether batch, real time, or both, some form of changed data capture can optimize synchronization.

Monitoring

Once the new platform has live data and active end users, monitor it to gauge success and to prioritize areas of needed improvement. Monitoring may count the number of end users (to measure adoption), outages (to quantify high availability), data defects (to ensure users are using

the new application correctly), and scalability metrics (number of transactions, data volume, concurrent end users, etc.).

Retiring the legacy platform

Don't burn the bridge too soon. A few data migrations fail in the first attempt, so the old platform should be left fully operational to accommodate a possible roll back. In most migrations, the legacy and new platforms must run concurrently for weeks or months as phases of the migration complete and are certified. In some cases, the legacy platform may be needed for financial closings or other business processes long after all end users and data have migrated to the new platform.

USER STORY
Lots of legacies linger longer than liked.

A few years ago, a company migrated its general ledger application from a legacy platform to a modern one. The project ran on schedule and on budget—until they began their first financial consolidation and closing on the new system. Several business processes were so tightly wedded to the older application (especially those involving multiple subdivisions) that they didn't work on the new one. To close the books on time and accurately, they migrated all current data back to the legacy platform. Until they could correct the situation, the firm's users continued to use the new general ledger application, though periodically migrating data back to the legacy platform for certain closing and financial reporting tasks.

Conclusions and Recommendations

In summary, many companies "do more with less" by improving and simplifying preexisting enterprise data architecture through data-movement techniques like the migration, consolidation, upgrade, and integration of databases and applications. Note that these are all commonly executed with data integration technologies like ETL, replication, and hand coding. Some projects require more than one of these data-movement techniques and technologies. Yet, regardless of the type of data-movement technique selected for a project, data migration is commonly required.

Data migration requires tools and technologies for data integration, profiling, and quality.

- **Use ETL-based tools for most data migration solutions.** ETL is the preferred technology because it easily handles migration's extreme transformations, large datasets, and iterative design. Even so, replication and hand-coded solutions are useful, too. EAI is not appropriate.
- **Profile source data carefully.** Otherwise, suffer unpredictable problems during development and deployment. For greatest productivity, profile with a tool, instead of using manual methods.
- **Expect data migration—like all variants of integration—to expose data quality issues.** Some of these are problems requiring a fix, while others are opportunities for enriching data. Either way, raise the bar to improve data and metadata as you migrate them.

Using an iterative method, develop and test data models, data maps, transformations, data flows, and stored procedures.

- **Use an iterative process to develop a data migration solution.** Expect solution design to be multi-phased for both technology deliverables and rollout to end users. Look for tools that support iterative development and deployment.
- **Allocate considerable time to modeling the target and mapping data to it.** Data modeling gets more complex as more data staging areas are added. Data mapping is harder than it sounds, because it includes mappings, data transformations, and the rules that control branching in data flows.
- **Don't overlook stored procedures and other in-database procedural logic.** These require migration, too. How (and whether) these are converted affects the whole target application, so base this decision on a consensus of database, application, and business people.

- **Decide what size and content of source data to test with.** Test with complete source datasets when they are modest in size and are fully useful off of their original platform. When source datasets are large, it's more efficient to work with a small-but-representative sample.

Expect to support redundant platforms during the migration and to face legacy retirement issues later.

- **Keep the project's momentum rolling after deployment.** After all, you still have to transfer the new platform to its new owners and administrators, synchronize data between the old and new platforms, and monitor the new platform to measure its success.
- **Expect that redundant instances of platforms will be required.** That's because old and new platforms must run concurrently while all phases of the project complete. For accurate but noninvasive testing, you may need to recreate source or target environments within your test environment. Plan accordingly to ensure you have the appropriate hardware and software.
- **Don't be in a hurry to unplug the legacy platform.** It might be needed for roll back and for the last groups of end users to be rolled over. And some legacy platforms are subject to lease and depreciation schedules, so check these before setting a date for legacy retirement.