UCT
SMART TOOLS. SMART CONVERSIONS.

Why data migrations often fail

Data migrations often fail because they are attempted with traditional methods

Custom programs and packaged ETL solutions are the most common methods used to migrate large volumes of data. Both of these efforts require some degree of development. In the case of a custom program, the entire solution is developed from the ground up. In the case of an ETL solution, the various interfaces must be created and then transformation logic is added to them. Both of these activities are typically completed using traditional software development methodologies. But traditional methods often fail when they are applied to data migrations. Why?

They are driven by people who lack system and business knowledge

Business analysts have the system knowledge but in a traditional development model, vendor or in-house programmers, who lack the necessary knowledge, drive the solution.

They depend on precise upfront requirements that cannot be attained

Often the source system is old, large and complex and the skill sets needed for requirements analysis are in short supply. It is very common that system documentation is poor, outdated, or missing. Some requirements simply come from knowledgeable people who have done migrations before but because data migrations are an uncommon event, resources with this skill set are unavailable to contribute critical requirements. Because of these things, many requirements are discovered mid-stream.

They result in an endless, unpredictable and unmanageable requirements refinement loop

This occurs because when new requirements are discovered in the middle of the lifecycle, traditional methods cannot easily accommodate them. The new requirements impact the design and implementation. The new changes to the implementation and additional testing uncovers more requirements. The new requirements impact the implementation and so on. This loop is expensive and hard to manage with traditional methodologies.

Data migration requirements are often vague or incomplete

Usually data migration requirements are captured in data dictionaries in table or spreadsheet format. A strong standard is seldom used to ensure that the requirements are complete, clear and unambiguous. Vague and incomplete requirements lead to implementation errors.

They don't allow the data migration requirements to be co-evolved with target system functional modifications

There is a natural codependency between the data migration and the target system functional modifications. Real production data, which is only available from the data migration efforts, exposes new requirements that impact the target system modifications. Target system modifications expose new requirements that impact the data migration. This results in each group waiting on the other to complete so they can move forward (often with a lot of finger pointing).

They don't provide tools or methods for identifying or measuring success

Each attempt at migrating data has some things that are good and some things that aren't. With large data sets and no tools, it is hard to easily and clearly identify what is good and what isn't.

The data migration design documentation can't be used to support the walkthroughs that are a common part of the process

It takes significant manpower to keep manually maintained documentation in synch with emerging requirements. As a result, manual documentation typically gets more and more inaccurate as the lifecycle progresses. Automatically created documentation, on the other hand, often contains raw code or some other unusable content or is presented in a format that is not usable. Inaccurate or unusable documentation reduces the effectiveness of walkthroughs.

UCT

SMART TOOLS. SMART CONVERSIONS.

ETL solutions have additional problems

Since ETL interfaces are often developed using the same techniques that drive custom program development, they fail for the same reasons. But ETL solutions are affected by additional issues that also contribute to failure. Why do ETL solutions fail when applied to data migrations?

Their various interfaces are developed using the same failure-prone traditional methodologies

These are discussed above. When these techniques are applied to data migrations, even those attempted with ETL tools, they can lead to failure.

They do not embrace legacy technologies

ETL solutions cannot utilize legacy functionality like system I/O routines that extract or load data or data conversion routines like functions that perform proprietary date calculations. In addition, they do not handle legacy or proprietary data storage formats that are common in older applications.

They require additional costly infrastructure

Various kinds of "middleware" products are available that allow legacy platforms to seamlessly communicate with newer platforms. This kind of middleware is often required to enable an ETL solution to access legacy files and databases. But this is a costly component to add to a system that may be on its way to being decommissioned and it requires a particular expertise to use.

They result in a mixture of legacy and modern components with manual steps in between

Even when the right middleware is available, older file types and proprietary data formats often necessitate that legacy-side components be developed using a different language and technology to extract data. The mix of technologies is a complex solution that requires multiple skill sets, multiple development environments and more or larger teams. This increases communication paths, implementation hours, cost and the probability of failure.