UCT
SMART TOOLS. SMART CONVERSIONS.

Why UCT's approach succeeds

UCT succeeds using knowledgeable experts, a proven methodology and a custom toolset

Dan O'Hara founded UCT in 1992 after spending many years developing custom data migration programs for vendors. He observed the issues that caused data migrations to fail and he specifically designed the UCT solution to resolve those issues. There were three key components to the UCT solution.

- **Knowledgeable people** - not just about financial systems but also knowledgeable about data migrations, which is a rarity.
- **A toolset that allowed the experts to drive the process**, eliminated unnecessary dependencies and communication paths, enabled the experts to respond rapidly to changing requirements and generated precise user-friendly documentation.
- **A methodology that could be flexibly adapted** to differing customer requirements and easily accommodated the new requirements that were regularly discovered late in the development lifecycle.

As technologies and the needs of the customer have changed, the UCT Data Migration Solution has been honed and refined to keep pace. One thing that has stayed constant is the people. UCT continues to employ the most knowledgeable financial system and data migration experts available.

Why is the UCT solution successful when other data migration solutions fail?

It is driven by the people with the system and business knowledge

Included in the toolset is DCA, which generates code directly from the data migration business logic. This logic is entered into DCA by business experts (not programmers) in much the same way that they would write data migration business rules into a spreadsheet or table. This empowers the business experts and completely eliminates their dependence on programmers.

It expects to discover and adapt to new requirements at any point in the lifecycle

DCA provides an environment that manages the business logic and allows it to be easily changed. Once changed, DCA rapidly generates new code that can be run immediately. To adjust the logic for a simple requirements change and execute the new solution only takes minutes. In addition, if metadata changes (like it can for a changing target system), DCA accepts the changes and directs the business expert's attention to them.

It minimizes the requirements refinement loop

The toolset handles new requirements easily (see above) and provides reporting tools for problem analysis. The methodology evolves the data migration solution repetitiously by slowly increasing the size and complexity of the test data set and accommodating new requirements. These make the refinement loop controllable and greatly shortens it.

It enforces precise expression of data migration requirements

Data migration requirements are captured in DCA. Once entered, they are validated to ensure they are clear, complete and unambiguous. The code generated from the requirements does precisely what is required and implementation errors are eliminated.

It allows the data migration requirements to be co-evolved with target system functional modifications

Since the toolset and methodology easily accommodate new requirements and changing metadata, the data migration does not need to wait for functional modifications to complete. Instead, data migration occurs in parallel with the functional mods and adapts to changes as they occur. This allows functional mods and data migration to each continue at their own pace.

It provides tools and methods for identifying and measuring success

This is accomplished by DCA Reporting. It introduces logic to detect and report common data migration errors in the generated code and lets the business expert introduce their own custom error detection and reporting logic. It collects all reported messages and formats them into easy to use reports that include analysis features that help the user identify error trends and root causes. It analyzes and summarizes message data to provide data migration success metrics.

It provides precise and usable data migration design documentation to support walkthroughs

DCA accepts business logic from the business expert and creates the data migration code from it. It creates the data migration design documentation from the very same business logic. This means that code and documentation are always perfectly synchronized. Since the business logic language is designed for non-technical users, it is verbose and easy to understand. Documents express the data migration requirements organized by table & column or record & field just like it is commonly expressed in data dictionaries. Documents are distributed in MS-Excel spreadsheets which are portable and easy-to-use.

The UCT Solution solves the problems that hamper ETL solutions

The UCT solution resolves the problems caused by traditional methodologies. It also resolves the issues that specifically impact ETL solutions. How does the UCT Solution address the issues that cause ETL data migration solutions to fail?

It uses a custom methodology and toolset

These are discussed above. The UCT Solution is specifically designed for data migrations, we've used it 100's of times, and we've always been successful.

It embraces legacy technologies

The UCT solution can 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, it handles legacy or proprietary data storage formats that are common in older applications.

It does not require additional costly infrastructure

The UCT Solution can run on any platform required and is a single platform solution. It does not require any costly "middleware" products or special expertise but can directly access legacy files and databases when the solution is executed in the legacy environment.

It results in a simple and elegant, fully automated, single-technology solution

DCA generates the data migration code in the same language and technology used for I/O code. There is no unnecessary mix of technologies. This elegant solution is simple and requires fewer skill sets, fewer development environments and smaller teams. This minimizes communication paths, implementation hours and cost and ensures success.