

UCT

Experience. Methods. Tools.

All you need for Success

Data Migration Planning

Data Migrations Projects

Release Upgrades

Trusted



UCT is the leading service provider for data conversion and integration in the financial services marketplace.

UCT was founded in 1992 by Dan O'Hara and has provided conversion support to many of the world's top financial services Companies.

Data Conversions and Migrations are our business!

We've seen it before

Our staff has worked on every vendor-supplied system in the Life Insurance marketplace, as well as home-grown systems. We have converted Life, Property & Casualty, Workers Comp, Health, court records, and other applications.

We can do it better

Whether you need a turnkey solution or just want help on one phase, UCT can provide powerful expertise.

- ▶ Project management
- ▶ Project planning and analysis
- ▶ Data auditing and cleansing
- ▶ Data conversion/migration
- ▶ Configuration support
- ▶ Data balancing
- ▶ Tools and tool training
- ▶ Programming
- ▶ Business support
- ▶ Technical support

Exceptional Results

At UCT, our focus is data migrations. Because of this, we differentiate ourselves in a number of ways.

- ▶ Deep insurance product understanding
- ▶ Superior custom tools
- ▶ Agile project approach
- ▶ Most extensive testing approach in the industry
- ▶ Proven delivery model
- ▶ Standards Based

UCT strongly supports industry standards like the *ACORD XML standard schemas.*

- ▶ Vendor & Tool Agnostic
- ▶ All Vendors
- ▶ All Platforms
- ▶ All Systems
- ▶ All Data Storage types

***Give us a try.
You'll be glad you did.***

Experience you can count on

A Different Model

Traditional software development methodologies fail when applied to data migrations.

UCT uses the Agile methodology to overcome the obstacles that are presented when little can be known about a source or target system. Using this methodology, our conversion solution is evolved through repetitive testing on increasingly larger data sets. This approach is flexible and anticipates discovering new requirements, which shortens the project timeline and maximizes conversion accuracy.

Data Conversions and Migrations are our business!

Our clients value our:

- ▶ **Expertise**
- ▶ **Experience**
- ▶ **Objectivity**
- ▶ **Concentrated Focus**
- ▶ **Results**



UCT is a proud sponsor of
Habitat for Humanity®

The Most Experience in the Industry

At UCT, our conversion experts have an average of more than 25 years of industry experience.

Experience and knowledge are the keys to a successful conversion and they are hard to find. At UCT, we have cross-system knowledge and programming expertise but the most valuable thing we provide is knowledge about the most successful ways to convert and migrate data gained through many years of experience. You need that knowledge for success:

- ▶ **Accurate estimates and timelines**
- ▶ **Experts that know systems and know where to look for customizations**
- ▶ **Conversion processes that have been thoroughly tested, used before, and are successful**
- ▶ **Migration experts that know what problems to anticipate, what workarounds are successful, and how to change data so it can be moved between systems**
- ▶ **Programmers that know how to extract and load data into your systems**
- ▶ **You need UCT**

UCT has successfully completed over 200 conversion projects. From that experience, we confidently predict that we will reduce your time frames and estimated costs by 40-60% and that we will successfully convert in excess of 99.9% of your data.

***Disciplined project execution.
Exceptional results.***

A Methodology that Works

Overcoming the problems inherent to Data Migrations

The UCT methodology is different because it emphasizes the role of the business analysts, clarifies and minimizes the role of the programmers, makes actual data visible early in the lifecycle, and automatically creates documentation.

Traditional software development methods do not work for data migrations

Business analysts, not programmers, have the real system knowledge. Because of this, a conversion without UCT results in a costly codependence between the business analysts and the programmers as they try to work together to evolve the high-level design, detailed-design, and implementation into a workable conversion system. If the legacy system is particularly aged then what was unknown about the system becomes all too clear when testing begins and the team must repeatedly return to the drawing board to try to resolve the new data problems.

UCT's methodology and tools turns the business analyst into a programmer

They record the business logic for the conversion in a 4GL language which is automatically turned into source code. They do not have to repeatedly return to the programmer for code modifications and improvements, which reduces the number of key personnel responsible for the development of a source code module. In turn, the programmer is allowed to focus on the I/O aspects of the conversion (or the Extract and Load portions of an ETL project), which attempt to reuse system-specific I/O APIs wherever convenient. *This segregation of responsibility reduces the required number of programmers and drives down costs.*

Making the data visible early is important to a successful conversion

In the UCT methodology, one of the first phases is data analysis and auditing. In this phase, the business analysts use reporting features and rules to create code that highlights data anomalies or any unexpected data characteristic. Since this is the first phase, data cleansing can be started immediately to "fix" data in the legacy system and the conversion maps and associated code modules can use the information to correctly handle all data permutations the first time. *Understanding how the system differs from your expectations and correctly coding things the first time drives down costs and shortens timelines.*

The methodology results in the automatic creation of documentation

At the end of a project, all of the business rules are encapsulated in a set of "datamaps". Each datamap handles the conversion of a subset of business data and is written in a 4GL language that non-programmers can easily read and understand (and create). It is invaluable for design walkthroughs, code walkthroughs, and post-project documentation. *Since the datamaps are used to automatically create code modules, the generated code is perfectly documented.*

Difficult problems call for Agile solutions.

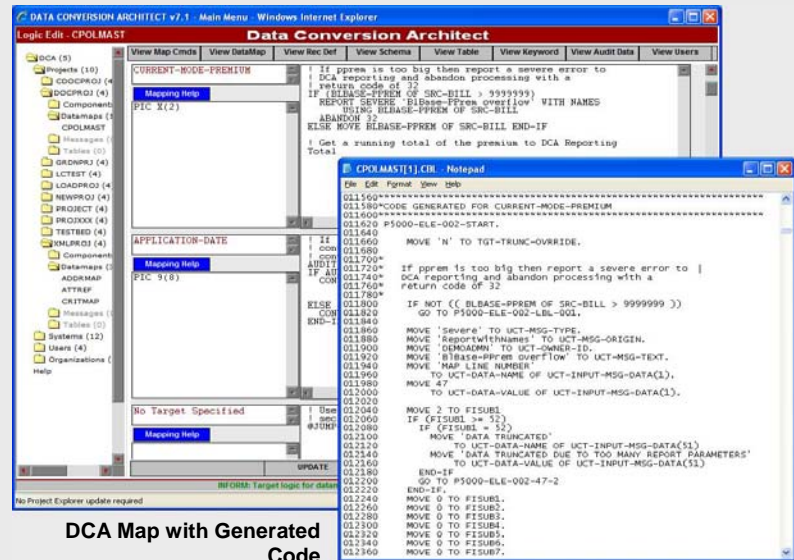
Data Conversion Architect

Tools that do the work for you

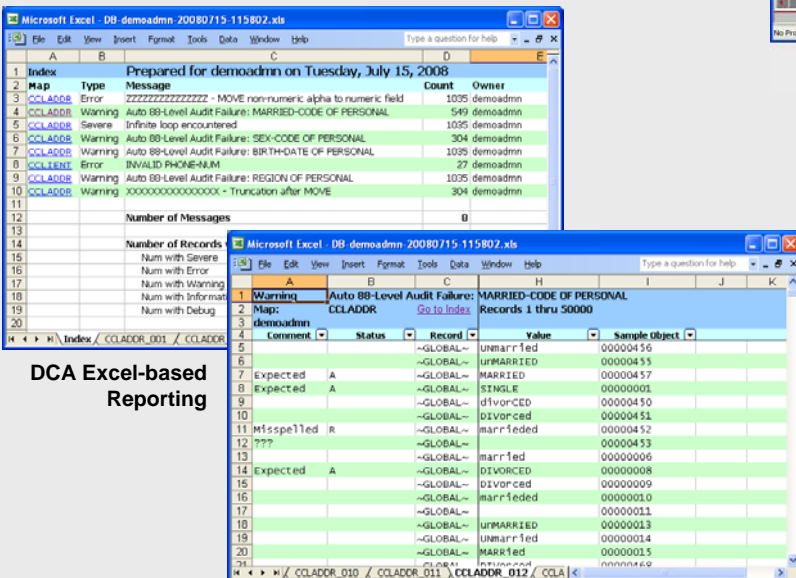
DCA is Central to the UCT Methodology

DCA complements the UCT Methodology by automating many common tasks. The tool's primary role is to allow team members to work together to collect simple business logic expressing auditing, conversion, and balancing actions. When the logic is mature, DCA turns it into complex source code that handles data type conversions, detects and reports errors, and is well structured and ready to be compiled and executed. One line of logic may generate many lines of source code and the logic serves as excellent documentation for walkthroughs, etc. DCA also includes:

- ▶ Automatic code generation
- ▶ Project Management features
- ▶ User Management features
- ▶ Full error reporting and analysis features
- ▶ Library of reusable code elements
- ▶ Database DDL processing services
- ▶ Schema-based XML stream creation features
- ▶ Complete auditing and balancing map
- ▶ Transaction creation and handling services
- ▶ Internal and external table management
- ▶ Conversion platform synchronization
- ▶ Project archiving and backup services



DCA Map with Generated Code



DCA Excel-based Reporting

DCA Reporting measures progress. Code generated by DCA automatically checks for errors and reports problems. Error messages can be viewed and analyzed in either DCA's internal reporting service or its excel-based service. Each of these services presents messages in well-formatted reports and include the ability to "drill-down" into the data from project to map to policy to logic line. **Evaluating the success of your conversion has never been easier.**