Data: The Neglected Key to a Successful Integration Effort

Utility Automation
Innovative Energy Solutions

By Mark Brulé

As the U.S. power utility landscape continues to change through mergers and acquisitions, and as utilities strive to improve efficiency and do more with fewer people, the demands placed on information systems will continue to increase. Desktop computing power has grown remarkably over the last decade. It is now possible to run sophisticated applications on the desktop that previously would have required a mainframe to support. This means that the tools available to planners, designers, mappers and others are far more capable than they used to be.

When a system integration takes place, systems are not the only resources to be integrated. Just as important is integrating the data that drives those systems.

Sophisticated applications have an appetite not only for raw computing speed and mass storage (things in great supply) but also for more developed data sources (things in great demand).

Consider the age-old database example: the telephone book.

If a telephone book database contains only names and numbers, then one can write an application that looks up a phone number based on a partial match of the name, or the reverse lookup of a name given a number. But that's about it. If an address field and a street map are added, then one can start plotting where the owners of the phones are, or provide a summary by ZIP code to provide lists for targeted marketing. The point is, the more sophisticated an application, the more data it requires. And if the application is expected to produce reliable results, then the data quality had better be high.

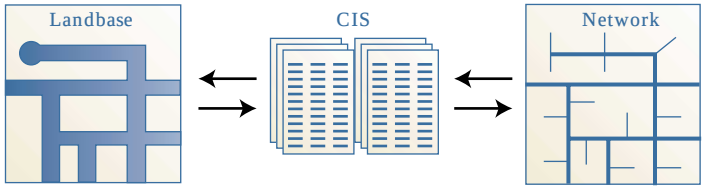
Not just data, but high-quality data is essential to utilities today. Providing good data, however, is more difficult than it sounds.

Where Data Problems Originate

Disparate data sources—one root cause of data-quality issues—are a frequent byproduct of corporate mergers. When two utilities merge, they typically have different systems for managing network data, customer data, loading information and so on. Even if they use the same systems to store such data, the implementations from the two companies can be vastly different.

Mapping systems are perfect examples of the kind of differences that can occur to make separately implemented systems incompatible. One

Data Integration



Tight integration of data from diverse sources is crucial to the successful implementation of many modern systems, such as outage management.

company can map its entire network in a tiled map system; another may have separate views for primary and secondary networks, or perhaps only operating maps. Even when the same mapping system is used to store and manipulate such data, merging the two data types can be a challenge. Devising a business process whereby the data is maintained properly can be equally difficult.

Additionally, utilities often have islands of information that were never meant to be integrated but that need to be integrated for special purposes. The most prevalent example involves the customer information system (CIS) and the mapping system. Originally intended to display a network infrastructure, mapping systems typically do not store customer data. The CIS, usually used for billing, is critical to the business as a means of being paid for its product. It is run by a different department (usually IT), and access to its data is often severely restricted.

Deploying an application such as outage management, however, is practically impossible without bringing these data sources together.

Integrating Data from Different Sources: An Example

A mapping system and a CIS can be integrated in a number of ways. Two approaches will be discussed. Suppose that the network data is held in a mapping system, whether that be a CAD system (e.g., AutoCAD or Microstation) or a GIS (e.g., Arc/Info, FRAMME or Smallworld), and that the CIS runs on a relational database (e.g., Oracle or SQL Server).

The goal is to integrate data from the two systems—as opposed to having one system consume the other's data and functionality. The philosophy behind integrating data from the systems is to recognize that the existing platforms have value and are probably well-suited to solve their respective problems, and that to combine the systems would require existing business processes to be re-engineered and personnel to be retrained. To accomplish this type of data integration, key information will need to be added to one system to indicate linkage to the other system.

In the network map and customer information example, one approach is to add a component ID to the CIS to indicate what device serves the given customer. This usually takes the form of the ID of the transformer to which the customer is attached. When populating the data into a target application, this gives a "location" on the network where the customer should be placed. For outage management and engineering analysis applications, this level of connection is usually sufficient.

The other approach is to add a service point to the network data, relate it in some way to the serving line or device, and store a premise ID or account number on the service object. When the service point is connected to the secondary network, this can ease the reconfiguring of customers that occurs when the secondary network is altered, but also means that more work must be performed when moving the data into the target application.

After choosing one of these integration approaches, the coding for maintenance applications and business process development can take place. But the question remains: How will the data be integrated? Without properly populating the data fields that are added, how reliable will the target application ultimately be? In this example of customer-to-network linkage, numerous approaches are typically considered. The CIS generally carries one form of location information known as the address. Unfortunately, this rarely translates into an easy and reliable point for connecting the customer to the network.

Linking Customers to the Network Accurately

Invariably, the first attempt at linking customers to network service points involves first "geocoding" the customers and then selecting the nearest transformer. Geocoding a customer involves assigning an x,y coordinate from the address, given a street landbase meant for this purpose. This allows the customer to be placed on the network map (assuming it is geographically referenced). Straight-line distances to nearby transformers are calculated and the closest is selected. Situations that make this approach dangerous include:

- multiple circuits involving a line of poles that carry more than one primary circuit;
- discrepancies between the landbase used for geocoding the customers and the landbase on which the network maps were developed;
- the presence of secondary networks that serve the low side of the transformer to several neighbors down the street;
- discrepancy between the address in the customer database and the street names in the geocoding source; and
- addresses in the customer data that are not found in the geocoding source's address ranges.

While this approach is readily implemented, experience shows that it successfully links a mere 25 percent to 50 percent of the customers to the proper transformer, and only 50 percent to 75 percent of customers on the proper primary circuit. This is woefully inadequate for driving most applications. Even the most sophisticated inference engine in an outage management system cannot deal with data that is this inaccurate.

What is required is a more sophisticated approach to integrating the data. Integrating the two systems is a straightforward technical problem that any good IT person can solve. Effectively integrating the data, and coming up with a business process for maintaining it, is a far more challenging task whose solutions vary greatly from one instance to the next. In confronting this task, two data-management strategies come to mind.

Middleware and Gateway Solutions

In the past, when data needed to be moved from one system to another and

possibly manipulated to fit into the target system, developers had one option: custom-coded solutions. While starting from scratch for each project meant that similar problems had to be solved multiple times, custom development produced the most efficient system for moving the data.

After a decade or two of lessons in the value of code reuse, products have been developed to assist in this process. Generally, a solution built for a generic purpose and adapted for a specific problem is slower than a custom solution. But since processing power has increased, and since the implementation of some communications systems has become quite advanced, it is now possible to consider building scalable, production-ready systems on top of these platforms.

The idea of building a general framework for connecting different applications and data sources across the utility is taking shape as a bus architecture. The strategy here is a software bus that acts much like its hardware equivalent. Applications connect to the bus by means of an adapter, and the applications communicate information solely through the bus product. This architecture uses a publish-and-subscribe metaphor, where applications that want a particular type of data tell the bus that they wish to subscribe to the data type, and the bus makes sure they get the right data when another application publishes it. Several questions must be considered to make this vision a reality:

What messaging service should be used? An array of choices (such as the Java Messaging Service, or TCP/IP sockets) exist that vary in the speed and size of messages that can be sent. The size of the message supported will influence the way data is transported around the system and may place additional constraints on the architecture.

What implementation of the bus should be used? There are some excellent products on the market that hide the messaging architecture and provide the ability to monitor activity on the bus

The idea of building a general framework for connecting different applications and data sources across the utility is taking shape as a bus architecture.

as well as configure what the attached applications will do and how the business process works (e.g., IBM's MQ Series, Vitria's BusinessWare, and TIBCO's product line).

What data model should be used? This is crucial. For the bus architecture to work, there needs to be an agreed-on format for the data that is passed around. Common Information Models are now in development for both the transmission and distribution realms, and products (such as SISCO's UIB) are putting those models into practice.

All of this is great at getting the data communicated from one point to another, and the addition of an overarching model to make sure all the applications are speaking the same language makes the bus a feasible approach. However, the underlying concern about data quality remains. If an application requires a representation of the network with customers attached, it is important that the network's structure is communicated in a language that the application can understand (the common model).

However, the user still needs to be sure that the data sent to the application is correct. The bus architecture and other middleware products help communicate the data, but that is not a replacement for software that can integrate the data properly and produce quality results. Gateway solutions offer not only data transport but also data-integration and data-development capabilities. A configurable gateway solution allows import from a wide range of data sources, sophisticated data integration and validation, and export in the format required by the specific outage management, engineering analysis, or other enterprise system.

Valuable Tools for Integrating Data

The success of any system built from separate components depends on how well the data can be shared among applications. It is vital that the data be communicated in a timely fashion, but that alone is not sufficient. Components must be in place to effectively manage the quality of the integrated data. A special consideration for power utilities is the use of spatial data in solving problems. Yet spatial data is often the most difficult to manage.

Good data-development tools provide the ability to make the most of network data. Such tools should be able to detect errors (based on electrical properties of networks) as well as assist in integrating data from diverse systems. They should not only detect errors but also help automate corrections when possible, and show where problems exist if they require manual attention. In addition, the tool set should allow users to validate the integrated data to ensure that it meets the requirements of the target system.

The bus architecture now being developed needs a robust communication layer and a common model for the data. The same is true for data-development tools, including those that are integrated in a gateway solution. These tools need to communicate with a wide variety of products (covering both spatial and non-spatial data types) and allow users to easily model the data they will be dealing with. This ability is required to help users deal effectively with the different types of problems and to develop software and processes for dealing with the data. ■

Mark Brulé is chief technology officer at Coherent Networks Inc.

Coherent Networks, Inc.
One Adler Drive
East Syracuse, NY 13057
Phone: 315-433-1010
Fax: 315-433-0070
www.coherentnetworks.com