Herding cats - the challenge of data and application integration

Clay Harter, chief technology officer, OpenSpirit Corporation, explains how the 'middleware' solution to application integration has taken shape to manage data across multiple data repositories

The upstream oil industry is highly data-intensive, and oil companies face a variety of challenges in their quest to effectively manage data across multiple data repositories. Focusing just on the geotechnical realm, companies have many projects containing data of various types—engineering, geologic, and geophysical data—and a host of issues arises as a result of employing the different applications that need to utilize this data.

Let's consider the case of Jack, a hypothetical geologist in an oil company who needs to run a reservoir characterization program on his PC desktop computer to build a reservoir model covering several offshore lease blocks. The program needs to integrate diverse data types, including seismic data and its interpretation, well information, logs, and formation tops. Jack doesn't know exactly where all the data he needs is located, so he asks his IT specialist, Jill, to help him locate and load the required data.

Jill will have to determine which projects have data covering the lease blocks of interest and then export data from these projects into appropriate exchange formats (e.g., SEGY for the seismic data, LAS for the well logs, and ASCII for the horizons and faults), then FTP the exported data to the PC, where Jill, perhaps with assistance from another IT specialist, has to run the data loaders for Jack's application. Additionally, Jill needs to make sure that all the data is in the same coordinate system and units before loading the data . Once all these tasks are complete, Jack is ready to roll.
Elapsed time: two weeks…minimum.

Jack's workflow, while not out of the ordinary, illustrates many of the software integration problems that companies have faced ever since this industry first started doing computer-based analysis and interpretation. As a community, we have tried various approaches to address these problems with limited success. Only recently has a solution appeared that allows effective management of diverse data in multiple repositories and in addition offers end users a window to innovation in allowing flexible multi-vendor application interoperability.

Problem #1: the application cannot readily access the data

Traditionally, when a company buys an application from a particular vendor, that application comes with its own data format, data model, or data repository. As the firm acquires applications from different vendors or even the same vendor, it often ends up with a diverse array of different databases or formats. This creates a number of integration problems. For example, a user may not know what projects exist because applications have difficulty discovering what projects exist outside their own project repositories or across multiple project catalogs. Some applications don't even have the notion of a catalogue of projects.

Even when the user can discover projects, he or she may run in to a problem if the application can't read a different data format or a different data model. In today's heterogeneous computing environment, where we have some applica-

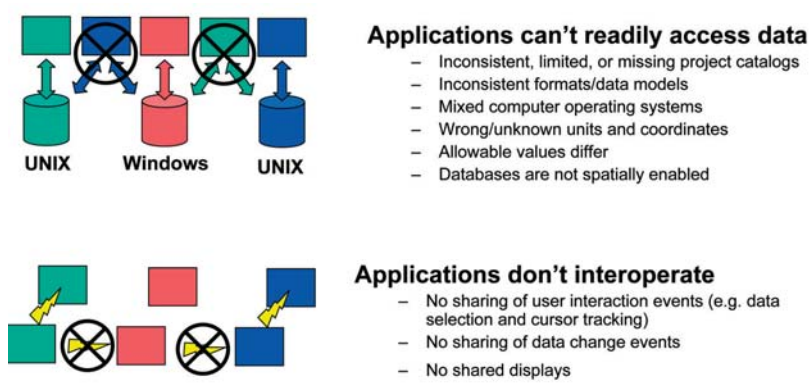

Figure 1 *Key Integration Problems*

tions on Unix (typically geologic and geophysical applications) and others on the PC Windows desktop (engineering and business productivity applications), we have the added complication of crossing operating system boundaries.

There are also subtler issues involved with reading data, such as differences in the coordinate system or the units between data sets, so a user trying to incorporate data into an application from other sources may not know how to interpret the spatial locations or measurement values. Also differing datastores may have varying allowable values for data attributes (e.g. one application may want to hear 'P&A' while another needs 'Plugged and Abandoned').

Another issue that arises is that as many project datastores are not spatially enabled, there is no efficient means of spatially selecting data, one of the easiest ways to select data.

Problem #2: applications will not interoperate outside their own suites

The other category of integration problem that companies face when they use a collection of applications is when applications won't interoperate. One can buy applications from a particular vendor and within that vendor's suite of applications there may be a notion of inter-process communication for the sharing of user events. Examples of user events include the sharing of data selection, cursor location, or data change events. By sharing these user interactions between applications a user can use a set of applications together as a 'virtual' application. Actions taken in one application cause a coordinated reaction in another. Applications from Landmark's OpenWorks or Schlumberger's GeoFrame application suites exhibit this type of event sharing within their own respective suite. There is no sharing, however, between different vendors applications and many vendors' applications have no notion of user interaction events at all. So this means that applications don't, in general, interoperate.

Approaches to problem resolution

As illustrated in Figure 2, our industry has tried a variety of approaches to address these integration problems. One approach, as embodied in the mission statements of POSC and PPDM, two not-for-profit organizations, is to specify one unified data model for databases so that different vendors could implement compliant databases and, therefore, applications. This has only been partially successful; it has resulted in a higher degree of similarity between vendors' products than what would otherwise have occurred, and there have been pockets of applications that do tie in to a common public data model. Generally speaking, however, efforts in this regard have not been as successful as was once hoped, for both commercial and technical reasons. There still are benefits to be gained from this kind of standardization, and the industry should continue to support these types of efforts. For example, standard sets of reference unit values are beginning to emerge as well as a common designation of coordinate systems, and these are influencing a degree of standardization in different vendors' products.

Another approach that companies have tried is the use of a single vendor to provide all of their applications, thereby delegating the issue of integration to that vendor. The vendor provides a suite of integrated applications that tie to a single project database and also provides inter-process communication messaging between those applications. There are now several vendors in the marketplace that offer suites of applications that go a long way toward fulfilling the needs of oil companies. However, most companies' needs extend beyond

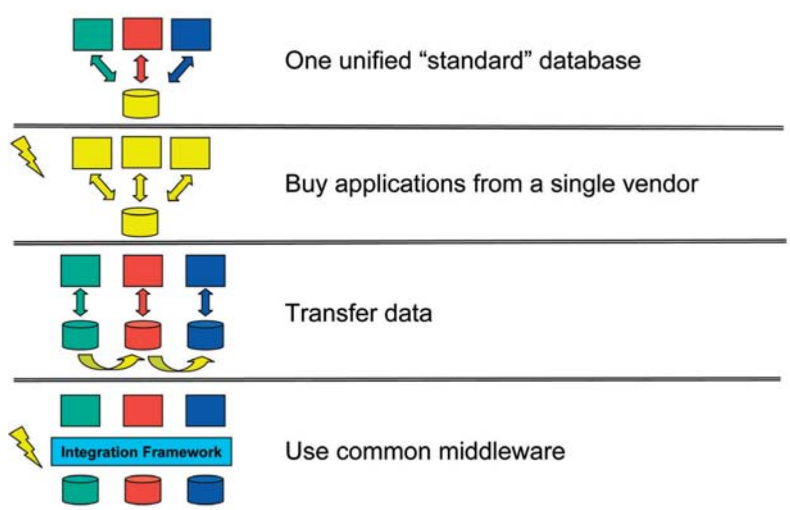

Figure 2 *Integration Approaches*

those that can be served by a single vendor. For example, a multidisciplinary asset team, with drilling engineers, petroleum engineers, reservoir engineers, geologists, and geophysicists all working together is unlikely to find a set of tools from one vendor to meet their collective needs. Because of this type of situation, even those companies that consider themselves a 'single vendor shop' end up buying applications from a variety of vendors, which once again puts the integration card back on the table.

In reality, the problem does not go away even with a single vendor. Application-suite vendors invariably grow by acquiring other software vendors, which can present integration issues until the new applications are integrated into that vendor's suite. Therefore, even though a company's applications may be coming from a single vendor, they probably won't continuously have the tight integration that is sought.

Another factor that companies often consider when deciding on whether or not to pursue a single-vendor approach is that application-suite providers tend to be large enterprises, and quite often in our industry some of the more innovative and potentially valuable solutions are coming from smaller companies. Oil companies that want to harvest the market for leading edge technologies want the flexibility of being able to incorporate different vendor products in their mix. Also, companies that may want to establish a single vendor relationship now also want the flexibility to change vendors in the future. They want to avoid 'vendor lock-in.'

These are some of the reasons that companies inevitably bring in applications from different vendors or applications from the same vendor that are not integrated. And once there are unintegrated applications, along with the separate databases coupled to them, companies have to routinely transfer data for use by different applications.

OpenSpirit—six years on

Development of the OpenSpirit Application Framework began in 1997 as a project funded by the OpenSpirit Alliance, a consortium comprised of oil companies and software vendors seeking to address inefficiencies and high costs related to the poor integration capabilities of E&P applications and their related data stores.

With research and development funded by consortium members, a multidisciplinary team of technologists delivered a prototype of an application integration framework in 1999. While the software met with great support among consortium members and the E&P community as a whole, it became apparent that in order for it to best serve the industry, it would need to be offered by a commercial entity responsive to market needs and not guided by a consortium.

The OpenSpirit Corporation, based near Houston, Texas, began operations in July 2000 as an independent software company. An off-the-shelf, standards based middleware product, the OpenSpirit application integration framework allows interoperability between multiple vendors' applications and data. To date, over 25 software vendors have licensed the OpenSpirit developer's kit. Additionally, there are now over 600 oil company end users in over 40 countries taking advantage of the framework to speed up critical workflows and enhance analysis in the geotechnical space, with more adopting the solution every day.

Putting the data frogs in a wheelbarrow

Data transfer is a 'time-honoured tradition' in our industry that involves the (expensive) development of the tools to do the transfer and the (even more expensive) use of those tools by specialists. The actual transfer is only one level of challenge; there is also the problem of keeping the original and replicated data updated, which may require some kind of two-way synchronization. There is a whole set of issues surrounding the very high cost of managing replicated data, but it is something that every company has to do. Many oil companies end up writing their own scripts and their own data transfer utilities.

There are times when there is no alternative to data transfer, as is the case for a mobile laptop user working off-site who needs the data on the computer's local drive, or when very high performance visualization tools are being used that require the data to be local in order to work optimally. The challenge in these cases becomes how to make the transfer easily and then how to provide some sort of data synchronization.

One solution for all the problems

The most recent approach to integration pulls apart the coupling between the application and the database by introducing a 'middleware' or 'integration framework' that allows applications and databases to work within that framework and not have to be tightly coupled. The advantage of this approach is that it allows oil companies to pick data management solutions, the databases, independently from their application choices.

With this approach, an application can work with more than one project data repository, even simultaneously. In an integration framework, the responsibility of discovering data and accessing data is delegated to the middleware. This middleware can also provide tight integration in terms of interoperability and sharing of user interaction events, which is not available in the data transfer scenario or with the common data model.

Let's go back to Jack to give an example. Running his reservoir characterization software with an integration middleware product, his application can issue a query to discover all the wells within a given lease block, regardless of what (or what type of) projects they might be in. Further, Jack's PC-based application is able to access data sitting on a UNIX file system without the necessity of having to share the file systems across operating systems. The middleware can deal with the required distribution and access of the data along with the appropriate authentication because it sits on top of the authentication mechanism the company already has in place with its existing project databases. Then, as the PC based application reads the desired data, it is transformed into the desired coordinate system and units and is presented through a common data model view. Jack can even select the data he wishes to use in another application, perhaps in his GIS map window, and his PC based reservoir characterization application will respond to this data selection event.

The bottom line is that Jack can go out and discover data in all the repositories and he can select the wells, seismic, and interpretation data he wants directly from his desktop, and make immediate use of it. In addition, Jill doesn't have to spend her time reformatting and moving data and can devote more time to maintaining the quality of the data.
Elapsed time: Five minutes.

Essential features of an integration framework

In order to provide the functionality illustrated in our anecdotal story about Jack and Jill, the integration framework must provide the following functionality:

- Ability to connect to multiple projects of potentially of varying type
- Metadata about project catalogs to allow applications to discover all projects
- Query service against a common data model
- Units conversion
- Coordinate transformation
- Reference value mappings to account for varying allowable attribute values
- Ability for Windows clients to access UNIX based projects
- Cross platform event messaging system to allow sharing of user interaction events

An example of such an integration framework is the OpenSpirit Framework offered by OpenSpirit Corporation. Figure 3 represents the main architectural concepts that have been implemented (or are planned to be implemented) in this integration framework.

In this diagram there are two different types of public API s (Application Programming Interfaces) – those that a

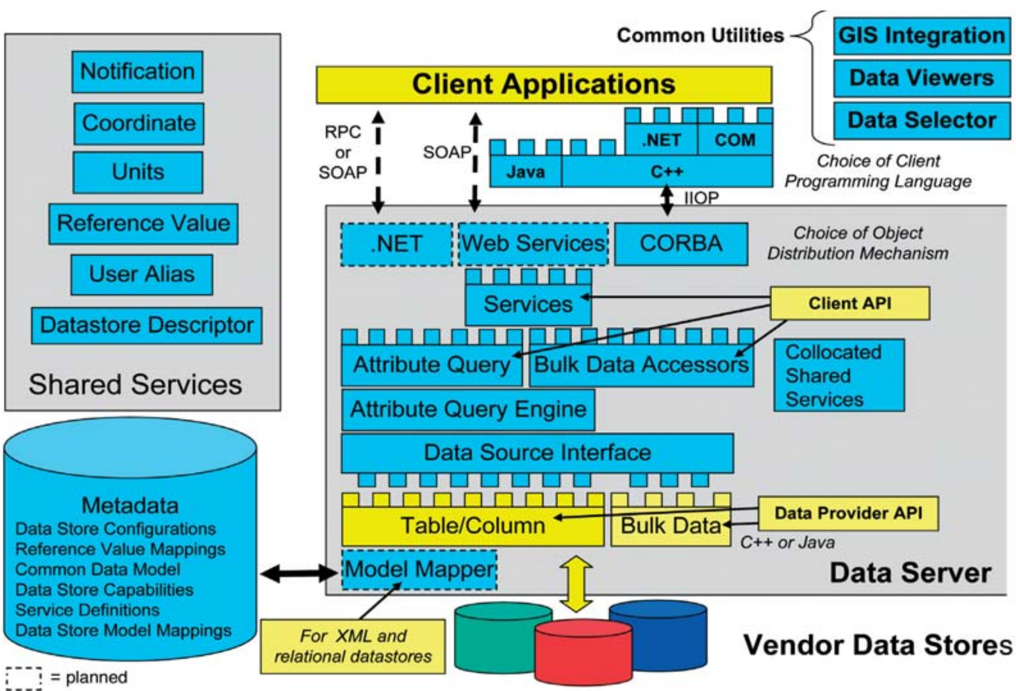

Figure 3 *OpenSpirit Framework Architecture*

developer uses if they wish to tie their application into the framework (the Client APIs) and those used by a developer who wishes to 'publish' their database or data to the framework (the Data Provider APIs).

Application integration

Client applications may discover data by either listening for data selection events or they may use the Attribute Query service to construct a query to discover and access data. The Attribute Query service uses a subset of SQL92 as its query language and allows queries to be issued across multiple projects (the client application sets the scope of the query) against a common data model as described by metadata contained in the metadata repository. The Datastore Descriptor service provides information on what datastores are available in a customer site and for a given store what projects are available. The Attribute Query service defines basic data types which include primitives such a string, float, integer, as well as point and arrays of these primitives.

A client application may also set a preferred coordinate system and unit set on the query and then all data will be returned according to these preferences. Most data types can be fully accessed through the create, read, update, and delete functionality offered through the Attribute Query service. Some data types such as seismic, stratigraphic grids, and horizon grids require specialized interfaces as provided through the Bulk Data Accessor interface. For example, to handle 3D seismic data four accessors are provided: a subvolume accessor, a planer accessor, a random trace accessor, and a random sample accessor. The service interface allows client applications to define and execute their own services within the data server. This would allow, for example, a developer to define a log resampling algorithm and to insert this into the OpenSpirit data server. This service interface has been used by OpenSpirit to define a copy service which allows client applications to cause data to be copied from one data server to another without requiring a client application to read or write any data.

There are a number of OpenSpirit Utility applications which may also be useful for applications developers and end-users. The DataSelector is a simple-to-use table view of all data types that allows a user to define and save queries and to send data selection events. There are three basic data viewers that are part of the framework including a well, section, and 3D viewer. These viewers may be used to preview data and also have the capability of sending and receiving data selection events. Application developers therefore have the choice of delegating data selection to these, or other, applications. In addition there are GIS integration tools which maintain a spatial catalogue of well, seismic, and interpretation data and then enable tools like ESRI's ArcView to send OpenSpirit data selection events.

Data integration

The Data Provider API consists of a generic virtual table interface and specialized bulk data accessors. In order to publish data into the framework a data provider simply implements the common table model (as described in metadata), in either Java or C++, against their underlying data or database. Bulk data interfaces are defined for data that is not well suited to a table model such as seismic data, stratigraphic grids, and horizon grids. The attribute query engine inside the OpenSpirit data server executes queries that can not be delegated to the underlying data store. For example, the OpenSpirit Attribute Query service defines a spatially between operator and in the event that the underlying data store is not spatially enabled the Attribute Query engine must handle this operation. In the event the datastore is spatially enabled, then it is preferable to delegate this to the implementation of the table/column interface.

If the underlying data store is a based on a relational database or XML file store then it is possible to have a generic implementation of the table/column model that is driven by a data model map stored in the metadata repository.

Conclusion

The quest for ready data access and application interoperability has been underway in our industry for 13 years. Various approaches to achieving these goals have met with varying degrees of success, but until recently there has not been a stand-out among the available solutions. Now, as many enterprises around the world implement the middleware approach, oil companies and E&P software developers are beginning to see the answer. With application integration middleware transparently connecting end users to the data that they need, and in the appropriate formats, valuable time and resources are shifting from data manipulation tasks to analysis and decision-making tasks—a shift that has compelling business benefits.