enterprise integrity

By DAVID McGOVERAN

Business Semantics

A few days ago, a colleague asked me what I thought was the most difficult integration problem. I answered "Business semantics," to which he replied "Meaning?" I couldn't resist: "Exactly!" Amusing perhaps, but a sad commentary on our sophistication.

Integration technology has come a long way in the past 10 years. We've developed highly scalable and distributed messaging architectures, implemented distributed directories and brokers, learned how to handle complex data transformation, begun to manage business processes, and even produced a few useful standards. It's all about being able to capture and move data between applications in a timely, robust fashion, while making certain that the target application can read the data produced by the source application. The technical problems involved in successful application integration and business-to-business (B2B) or even business-to-consumer (B2C) e-commerce can be reduced to three tasks:

- Connectivity
- Timely capture and purveyance of data
- Understanding the data.

Sounds a little simplistic, you say? To the contrary, it's quite complex. Let's examine these tasks one at a time (albeit at a high level) to see the origins of the complexity.

- Common plumbing, also known as a messaging architecture or transport layer, provides most connectivity. Add transformation and adapter technology, and the job is conceptually done. Of course, the devil is in the details. Considerable work may be required to bridge across various transports and to adapt between various data formats. Nonetheless, if the source and target formats are known, transformation is pretty straightforward.
- Timely capture and purveyance of the data is achieved by selecting a scalable architecture, making data capture event-driven, and then supplying sufficient network bandwidth and computational resources to do the job. Again, the devil is in the details. We must understand what constitutes events and what constitutes timeliness from a business perspective.
- Regardless of the installed technology, all is for naught if the captured data is not understood. We have to know more about the data formats to know how to map data elements in the source to data elements in the target. To put it another way, there's no point in mapping data elements unless the meaning and business use of those elements are consistent. This problem of identifying and maintaining consistent business semantics is indeed a tough problem.

The business meaning of a data element is defined by the ways in which it may be used, not in the abstract. Clearly, business rules and integrity constraints are intrinsic, essential aspects of business semantics. Even though interfaces may be documented, we rarely know the precise meanings of the data elements produced by an application. Even when we do, packaged application vendors make changes in data usage that, while presumed harmless, subtly modify business semantics. Giving a data element a label, as happens with XML, does not solve the problem. Furthermore, all changes in data usage (e.g., usage by additional applications or modification in usage) are semantic changes.

So, how do we capture and maintain business semantics? Strangely, this problem was solved long ago. It's called database design (and I don't mean so-called, entity-relationship modeling). Formal database design identifies the data types (which specifies legitimate values and operations), the relationships among those elements, and the legitimate operations on related sets of values. In the relational model, the complete set of operations is well defined, and a constrained subset preserves implementation-independent business semantics. Of course, all this is meaningless (pun intended) unless formal design practice is followed and documented. That means treating all public data elements pertaining to a set of integrated applications, whether stored in a relational database or not, as though they were attributes of a formally designed relational database.

I'm not suggesting that all data is "relational" (whatever that would mean). Nor do I suggest that formal relational design is the only method of achieving the goal of identifying and maintaining business semantics, thereby enabling proper mapping of data elements in an integration environment. Considering the accelerating erosion of database design discipline among vendors and practitioners (accompanied by complaints of how difficult formal design can be), it's ironic that so many integration difficulties arise due to unspecified or corrupted business semantics that must then be laboriously "patched." Data, and how it's used, forms the basis of all information processing. Surely maintainable business semantics — and the enterprise integrity that results from using a proven technique — is worth a little formal design effort involving application vendors and developers. eAI

David McGoveran is president of Alternative Technologies, Inc. He has more than 20 years' experience with mission-critical applications and has authored numerous technical articles on application integration. e-Mail: mcgoveran@alternativetech.com.