enterprise integrity

By DAVID McGOVERAN

Data Integration, Part III

If you've read this column the last few months, you know that data integration must address the difficult problem of rectifying possibly disparate meanings. You also know that, although integrity constraints capture operational meaning of data elements, we're rarely so lucky as to have an explicit repository. Implicit integrity constraints and the absence of a repository of explicit integrity constraints represent a serious cost problem for the data integrator. Few data integrators recognize that integrity constraints convey data semantics and even fewer actively investigate data semantics when faced with a data integration task. For those that do, two methods are dominant: interviewing experts and examining software systems.

Asking "someone who should know" the meaning of data elements initially seems a good idea. However, relying on business or IT domain experts through interviews is unreliable for several reasons. First, interviews are unlikely to result in a complete understanding of the data element in question. Having different experiences and views of the business, business domain experts seldom agree. Often, they're not able to communicate intended meaning in a way that a technologist can understand and they may omit important implications. The skills required to glean a clear understanding from an expert aren't easily acquired. Few have the skills necessary to rationalize seemingly conflicting definitions from multiple experts.

Available business domain experts probably had little influence on applications that manipulate the data element. Persons who have had no contact with the business domain expert typically design and write most applications. If the application was developed in-house, few IT departments have the luxury of permitting their in-house developers to work directly with such experts, nor can many business managers afford to contribute much expertise to the software engineering process. In the case of packaged applications, the business domain experts available to an in-house data integrator won't have influenced the package vendor's programmers.

Assuming the previous problems are overcome, many applications may manipulate a specific data element and so contribute to its operational meaning. With today's rapid turnover in IT, some of those applications are likely to have been developed by individuals no longer accessible. Even original business domain experts may be long gone.

Even if the data integrator has access to the application designer and programmer, human memory is fallible. Those individuals are unlikely to remember all constraints and assumptions about data element semantics that have been embedded in code.

The alternative to interviews — analyzing software systems — is a laborious, error-prone process. While mining application code for data semantics is possible, the common intertwining of business rules with procedural, physical resource-dependent algorithms results in both confusion and tedium. Few analysts can manage the challenge of separating coded constraints that impart business operational meaning from algorithm-specific artifacts in a particular portion of code. Even if they do, the analysis won't impart a complete understanding of the global uses of a data element or an accurate classification hierarchy.

Tracking down all the relevant code that implements constraints is certainly difficult; finding all the operational constraints when that code executes is simply impractical. Regardless of the deployment architecture, the scheduling and prioritizing of code execution often embody important business constraints. Such constraints impart an operational meaning to the environment of a particular business task, determining compatibility with other tasks. In so doing, data elements used in those tasks are effectively sub-typed.

Transactional constraints are among the more important constraints sometimes implemented external to application code. How transactions encapsulate constraints is a topic for a future column. Suffice it to say now that transactions bind specific uses of a data type together and are invoked only when particular circumstances exist — when certain integrity constraints are satisfied. Again, the meaning of affected data elements is effectively sub-typed.

When data elements have multiple, mutually exclusive sub-types, the data integrator must not combine their respective values, nor transform a value generated from one into a value intended for another. Contemplating this worst case scenario should convince you that data semantics is essential to data integration and can only rarely be met through traditional investigative methods. Indeed, one must use data semantics to construct enterprise integrity. 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; Website: www.alternativetech.com.