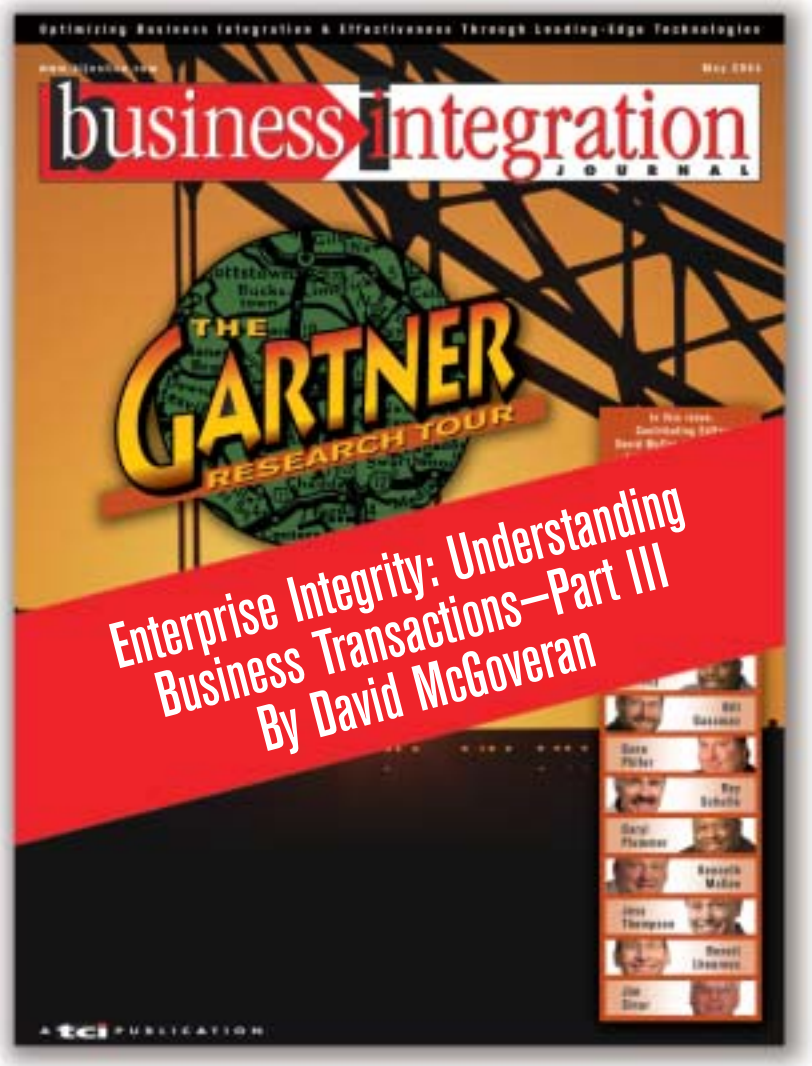
This article appeared in the
May 2004 issue of

business integration JOURNAL



Subscribe instantly at
www.bijonline.com

- Free in the U.S.
- $18 per year in Canada and Mexico
- $96 per year everywhere else

ENTERPRISE INTEGRITY

BY DAVID McGOVERAN

Understanding Business Transactions: Part III

In my last two columns, I've explained the fundamentals of physical and logical transactions, with emphasis on the ACID properties. As we've seen, if the operational properties of a software system aren't equivalent to strict enforcement of the ACID (Atomic, Consistent, Isolated, and Durable) properties, the system's computational output isn't trustworthy. Furthermore, these properties are essential to the formal definition of transactions. The relationship between formal transactions and business transaction (as commonly understood) cannot be explained except through these properties. In this and next month's columns, I'll discuss business scenarios analogous to ACID properties.

The notion of atomicity is pretty familiar in business, albeit slightly disguised. Every business is familiar with the concept of contractual commitment. A contract is, in essence, a specification of a business transaction that is either satisfied or breached when executed and, in this sense, is atomic. Some contracts specify rules of engagement for multiple business transactions between the parties. For many simple, or at least common, business transactions, contracts are implicit in common law. Of course, if a contract is breached, the parties may accept partial satisfaction in reaching a resolution, but the concept of breach treats the contract as atomic and so resolution is immaterial. In some cases, contract requirements may be renegotiated with each execution. This effectively changes the transaction definition and therefore exactly which set of actions is atomic.

The parties to business transactions can be defined in many ways. Examples include a business and any of its partners, suppliers, contractors, customers, stockholders, boardmembers, unions, or employees. The last requires a bit of explanation. An employee is under contract with the employer to perform a job. That job is understood through some combination of written procedures, verbal instructions, and the like, which—loosely—define the individual business transactions the employee may execute on behalf of the employer in exchange for pay.

The consistency conditions that a business transaction must satisfy on completion may, of course, be predefined and fixed, similar to those of a logical transaction. Alternatively, and unlike a logical transaction, they may be negotiated in the course of the business transaction or may be parametric. This flexibility is crucial, since the business state can be in rapid flux and not all conditions or requirements can be anticipated. Certainly there exist business processes for which most, if not all, possible events and conditions can be anticipated. Similarly, there exist business operations that can be constrained sufficiently so unanticipated variations are all but eliminated. One way of understanding the goals of Straight-Through Processing (STP) or Six Sigma is as an implementation of one of these two situations for a particular business process.

Logical transaction systems are usually semiclosed, meaning the types of external events that can influence them are limited and anticipated. By contrast, business transaction systems are usually open, responding to the influences of unexpected events. This fact has an important impact on how consistency is handled. Whereas the consistency conditions satisfied at logical transaction boundaries tend to be relatively unchanging and specific, the consistency conditions satisfied at business transaction boundaries may be variable and somewhat more general. They are context-sensitive and adaptive. Rather than a lack of precision, this provides business flexibility through abstraction and generality.

The consistency conditions applied to logical transactions are usually a set of integrity rules that can be jointly applied to the final state. In logical terms, they form a single integrity constraint by logical conjunction. By contrast, a business transaction may have multiple, alternative sets of integrity rules that can be applied to the final state to test for an acceptable outcome. In logical terms, they form a single integrity constraint by logical disjunction of those alternative sets. Some alternatives may be mutually exclusive.

A system state, event, or transition can always be represented as data, and so we often reduce consistency conditions to mere data rules. This is a simplification. Since business data semantics derive from the specific intent and utility of business operations, business changes can invalidate the data model representing the business. Transactions preserve business meaning only through carefully designed data models and consistent *operationally-based* data definitions—from source to application to database. Understanding consistency conditions as mere data rules is an error. The error is worse for business transactions because they often incorporate business operations, events, processes, and situations not yet represented as data. Until we return to this topic, consider how well transactions and data represent the business *integrity* in your *enterprise*. **bij**

About the Author

David McGoveran is president of Alternative Technologies. He has more than 25 years of experience with mission-critical applications and has authored numerous technical articles on application integration.
e-Mail: mcgoveran@bijonline.com
Website: www.alternativetech.com