Full circle for Web services transactions?

Submission for High Performance Transaction Systems Workshop, October 2003

Mark Little,
Chief Architect, Transactions,
Arjuna Technologies Ltd,
(*mark.little@arjuna.com*)

Introduction

The concept of atomic transactions has played a cornerstone role in creating today's enterprise application environments by providing guaranteed consistent outcome in complex multiparty business operations and a useful separation of concerns in applications. While numerous multiparty business applications involve various patterns based on atomic transactions in order to solve non-trivial business problems, it was not until recently the word "business transactions" accumulated any concrete meaning. Rapid developments in Internet infrastructure and protocols has yielded a new type of application interoperation that makes concepts which could only previously be considered in an abstract form an implementation reality. The effects of such changes have been felt most strongly in business environments, fuelling the mindset for a transition from traditional atomic transactions to extended transaction models better suited for Internet interoperation.

Most business-to-business applications require transactional support in order to guarantee consistent outcome and correct execution. These applications often involve long running computations, loosely coupled systems and components that do not share data, location, or administration and it is thus difficult to incorporate traditional ACID transactions within such architectures. For example, an airline reservation system may reserve a seat on a flight for an individual for a specific period of time, but if the individual does not confirm the seat within that period it will be unreserved.

In 2001, a consortium of companies including Hewlett-Packard, Oracle and BEA began work on the OASIS Business Transaction Protocol (BTP), which was aimed at business-to-business transactions in loosely coupled domains such as Web Services [1]. The specification developed two new models for "transactions", requiring business-level decisions to be incorporated within the transaction infrastructure. By April 2002 it had reached the point of a committee specification. Hewlett-Packard quickly followed this with the first commercial product based on this specification.

However, notable by their absence from BTP were Web Services heavy weights IBM and Microsoft, who in August 2002 released their own specifications: Web Services Coordination (WS-C) [2] and Web Services Transactions (WS-T) [3]. Although the WS-T specification also defines two "transaction" models, they are both very much grounded in existing transactional infrastructures. In particular, business applications would use WS-T in the same way applications use traditional transaction systems; there is no attempt to force this business logic into the transactional infrastructure.

Unfortunately for developers who require transaction support in their Web Services, the release of two competing specifications has effectively frozen the market. Is this a case of the "not invented here syndrome" (sometimes known as "my protocol is better than yours") or is there some fundamental reason why two different protocols have arisen? Have requirements really changed so much in the last two years?

Why two protocols?

In order to answer this question it is first necessary to give some (brief) background on what each specification provides and what has driven its development.

OASIS BTP

To ensure *atomicity* between multiple participants, BTP uses a *two-phase completion protocol*: during the first phase (*prepare*), an individual participant must make durable any state changes that occurred within the scope of the transaction, such that those changes can either be undone (*cancelled*) or made durable (*confirmed*) later once consensus has been achieved. Although BTP uses a two-phase protocol, it *does not imply ACID semantics*. How implementations of the prepare, confirm and cancel phases are provided is a back-end implementation decision. Issues to do with consistency and isolation of data are also back-end choices and *not* imposed or assumed by BTP.

Because the traditional two-phase algorithm does not impose any restrictions on the time between executing the first and second phases, BTP took the approach of using this to allow business-logic decisions to be inserted between the phases. What this means is that users have to drive the two phases explicitly in what BTP terms an *open-top completion protocol*. The application has complete control over when transactions prepare, and using use whatever business logic is required later determine which transactions to confirm or cancel. Prepare becomes part of the service business logic, for example.

BTP introduced two types of *extended transactions*, both using the open-top completion protocol:

- *Atom*: The outcome of an atom is guaranteed to be atomic;
- *Cohesion*: this type of transaction was introduced in order to relax atomicity. The two-phase protocol for a cohesion is parameterised to allow a user to specify precisely which participants to prepare, cancel or confirm.

WS-C/T

The fundamental idea underpinning WS-Coordination is that there is a generic need for a coordination infrastructure in a Web services environment. The WS-Coordination specification defines a framework that allows different coordination protocols to be plugged-in to coordinate work between clients, services and participants.

Whatever coordination protocol is used, and in whatever domain it is deployed, the same generic requirements are present:

- Instantiation (or activation) of a new coordinator for the specific coordination protocol, for a particular application instance;

- Registration of participants with the coordinator;
- Propagation of context;

The WS-Transaction specification plugs into WS-C and proposes two distinct models, where each supports the semantics of a particular kind of B2B interaction:

- Atomic Transaction: is similar to traditional ACID transactions and intended to support short-duration interactions where ACID semantics are appropriate.
- Business Activity: is designed specifically for long-duration interactions, where exclusively locking resources is impossible or impractical. In this model services are requested to do work and may provide compensators to be executed if the business-to-business interaction aborts. How services do their work and provide compensation mechanisms is not the domain of the WS-Transaction specification, but an implementation decision for the service provider.

Comparing and contrasting

Although at first glance it may seem like there is commonality between the two specifications (both support a two-phase completion protocol, for example), there are significant differences from both a protocol and implementation perspective.

Pros and Cons of BTP

As you might expect from a specification that took over a year to develop, on the plus side the BTP specification is well formed and complete. Unfortunately, although the protocol is not complex to understand, the specification is nearly 200 pages! It is thus not an easy sell for customers or analysts (and sometimes implementers).

What does it mean to be a user of a Web services transaction? Initially it may seem like a good idea to let business logic directly affect the flow of a transaction from within the "commit" protocol, but in practice it doesn't really work: it blurs the distinction between what you would expect from a transaction protocol (guarantees of consistency, isolation etc.) which are essentially non-functional aspects of a business "transaction", with the functional aspects (reserve my flight, book me a taxi, etc.) In BTP, because business logic is encoded within the transaction protocol, it essentially means that a user had to be closely tied to the (or perhaps even be a) coordinator! Business information, such as the ability for a participant to remain "prepared" (e.g., hold onto a hotel room) for a specific period of time is propagated from the participant to the coordinator, but there is nothing within the protocol to allow this information to filter up to the application/client where it really belongs!

In fact, in order to use cohesions it is also necessary for Web services to expose back end implementation choices about participants: in order to parameterise the two-phase completion protocol, the terminator of the cohesion obviously needs to be able to say "prepare A and B and cancel C and D", where A, B, C and D are participants that have been enrolled in the cohesion by services (such as a flight reservation system). In a traditional transaction system users don't see the participants (imagine if you had to explicitly tell all of your XA resource managers to prepare and commit?) Naturally this is

something that programmers don't feel comfortable with and it goes against the Web services orthodoxy. Because BTP requires transaction control to use the "open top" approach, it is difficult to leverage existing enterprise transaction implementations.

Furthermore, the BTP specification expends great efforts to ensure that two-phase completion does not imply ACID semantics. This is good in so far that traditional ACID transactions are not suitable for all types of Web services interactions. However, everything is left up to back-end implementation choices and there is nothing in the protocol (implicit or explicit) to allow a user to determine what choices have been made. Therefore, it is impossible to reason about the ultimate correctness of a distributed application. For example, if you wanted to use BTP for ACID transactions, then of course services could use traditional XA resource managers (for example) wrapper by BTP participants. Unfortunately, there is no way within the BTP for those services to inform external users that this is what they have done so that they can safely be used within the scope of a BTP "ACID" transaction.

Pros and Cons of WS-T

Both the WS-C and WS-T specifications are smaller than BTP, at about 45 pages in total. It is apparent from the specifications that simplicity and interoperability with existing transaction infrastructures played a key role in their development. Unfortunately they are also incomplete and have several protocol errors. For example, although heuristic outcomes are inevitable in distributed transactions, no support is provided in WS-T. Likewise, distributed recovery is paid very little attention. However, these are all issues that subsequent revisions can obviously address.

On the plus side, the separation of coordination from transactions is good: coordination is a more fundamental requirement and a separate framework offers the chance for a cleaner separation of concerns [4]. Because WS-C does not imply transactionality or a specific protocol implementation, it can therefore be used in more places than other protocols that have use of coordination but are tied to transactions (such as BTP).

The fact that WS-T Atomic Transactions are meant specifically for closely-coupled interactions with ACID semantics makes integration with back-end infrastructures easier. Web Services are for interoperability as much as for the Internet. As such, interoperability of existing transaction processing systems will be an important part of Web Services transactions: such systems already form the backbone of enterprise level applications and will continue to do so for the Web services equivalent. Business-to-business activities will involve back-end transaction processing systems either directly or indirectly and being able to tie together these environments will be the key to the successful take-up of Web Services transactions. It also takes away any ambiguity from users and services: they know a priori what semantics to expect.

In the realm of "extended transactions", the WS-T Business Activity also plays very well. It gives service developers complete freedom to define compensation mechanisms that best suit their services (for example, using Atomic Transactions where necessary), whilst at the same time providing a simple model for the users of these services. In addition, it ties in well with Web services choreography techniques.

What does this mean for transaction vendors?

Two years ago the world of Web services and transactions looked like a new frontier, requiring new techniques to address the problems that it presented. BTP was seen as the solution to those problems. Unfortunately, with the benefit of hindsight it did not address what users really want: the ability to use existing enterprise infrastructures and applications and for "Web services transactions" to operate as the glue between different corporate domains. And it had better be simple to use and understand as well!

The BTP model is superior to WS-T in several respects, but crucially it does not address the issues of transaction interoperability: most enterprise transaction systems do not expose their coordinators through the two-phase protocol. In addition, BTP has many subtle (and some not-so-subtle) impacts on implementations, both at the transaction level but more importantly at the user/service level.

So does this mean that the answer to Web services transactions is what we have had for the past 20+ years but using XML and SOAP? Yes and no! If we had the luxury to start from scratch and force everyone to throw away their corporate investments in infrastructure, training etc., then something *based* on BTP *might* be the answer. In the real world, however, we can't do that and it is unreasonable to assume otherwise. The investment in transaction processing systems over the past few decades has cost $billions and any scheme to leverage that investment rather than replace it is the way forward.

Much has been made of the fact that ACID transactions aren't suitable for loosely coupled environments like the Web. However, very little attention has been paid to the fact that these loosely coupled environments tend to have large strongly coupled corporate infrastructures behind them! When BTP started, the question should not have been "what can replace ACID transactions?", but rather "how can we leverage what already exists?"

References

[1] BTP Committee specification, http://www.oasis-open.org/committees/business-transactions/, April 2002.

[2] Web Services Coordination Specification, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-coordination.asp, August 2002.

[3] Web Services Transactions Specification, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-transaction.asp, August 2002.

[4] "A Framework for Implementing Business Transactions on the Web", Hewlett-Packard initial submission to BTP, March 2001, http://www.oasis-open.org/committees/business-transactions/