Interposition, Web Services and the Babel[1] fish

Mark Little, Arjuna Technologies Ltd,
Eric Newcomer, IONA Technologies Ltd

Position Paper for HPTS 2003

Transaction processing is at the core of commerce. All businesses have the concept of a transaction, but realize the concept using various architectures. Some transactions are very simple, such as purchasing a book or transferring funds and can be processed immediately. Other transactions are more complex, such as fulfilling a purchase order or completing an insurance claim, and may take days or even years to process. Transaction processing forms the core of enterprise information systems and often drives the business. Once developed and proven, successful TP systems stay in place for decades.

Business-to-business applications occur between loosely coupled *business domains* that do not share data, location, or administration and have their own internal infrastructures, such as corporate workflow systems, asynchronous messaging, or a variety of transaction processing approaches etc. However, the overall business interactions (business process) will require some level of transactional support in order to guarantee consistent outcome and correct execution.

With the advent of Web services, these business domains are typically exposed to the world as individual services. Multiple Web services typically cooperate to perform a shared function, such as multiple related operations on a shared resource such as a database or display, or processing different portions of a purchase order using a predefined sequence.

A business process is split into *business tasks* and each task executes within a specific business domain. A business domain may itself be subdivided into other business domains in a recursive manner. An individual task may require multiple services to work.

Business transactions are responsible for managing interactions *between* these domains. All of these domains are *coordinated* to perform the overall business process, often in a workflow-like manner. Essentially some "rule engine" must direct, coordinate and monitor execution of tasks arranged to form the business process [1]

This is obviously an extremely important and evolving area of the software industry. How should these domains be tied together, when each domain may use a completely different internal protocol? Is there a Universal Adapter that converts from one domain specific protocol to another and coordinates interactions across domains to ensure consistency? What software is at the heart of the "rule engine" that directs the flow of execution of tasks?

---

[1] Thanks to Douglas Adams!

As an industry, software engineers are notorious for re-inventing the wheel. However, as an important part of that industry, do we as developers of transaction systems have to re-invent (or allow to be re-invented) reliable coordination, when we've already done this extremely successfully for existing enterprise transaction systems?

The transaction coordinator implementation at the heart of most industrial strength transaction systems will have been developed over many years and optimised for performance and reliability. Because of the places where transactions systems are used and the reliance companies place on them, the coordinator can be trusted to work correctly despite failures such as machine or network crashes, or other issues such as overloaded machines. This level of trust is one that does not typically extend to other types of software components.

There has been much discussion over the past two years on the fact that ACID transactions are not suitable for most Web services transactions. Whilst that is probably true, many people read that as "current transaction infrastructures are not suitable for most Web services transactions"; this is a different statement entirely and overlooks the important aspects of what constitutes a transaction processing system.

Luckily for us as developers of transaction systems and those companies who have invested large sums of money in transactional infrastructures, existing transaction systems can (and should) form the heart of Web services coordination and management. Although existing ACID transaction systems may be unsuited in their entirety for Web services transactions, the *coordination aspect* (the core of all transaction systems) can be used in isolation and is extremely important for this evolving area. What this means is that the development of Web services transactions protocols such as BTP [2] and WS-T [3] cannot (and should not) occur in isolation from existing infrastructures.

Cooperating Web services are called *participants* in a transactional unit of work. Participants are minimally identified as Web services that share a common context. A *context* is a data structure containing information pertinent to the shared purpose of the participants, such as the identification of a shared resource, collection of results, common security information, or pointer to the last-known stable state of a business process.

One of the main benefits of a *coordinator* is that it can take the responsibility for notifying the participants of the outcome, persisting the outcomes of the participants and managing the context. A coordinator becomes a participant when it registers itself with another coordinator for the purpose of representing a set of other, typically local participants. When a coordinator represents a set of local participants, this is called *interposition.*

Interposition assists in achieving interoperability because the interposed coordinator (our Babel fish) can also translate a neutral outcome protocol into a platform specific protocol. The main benefit of adding transaction-based protocols is that the participants and the coordinator negotiate a set of agreed actions or behaviours based on the outcome, such as rollback, compensation, three-phase commit, etc.

As far as the parent coordinator is concerned, the interposed (child) coordinator is a participant that obeys the parent's transaction protocol (typically two-phase). However, each child coordinator will be tailored to the domain in which it operates and the protocol(s) that domain uses. So, as far as the child's participants (and services) are concerned, it is a coordinator that executes their protocol (e.g., three-phase).

Prior to the arrival of Web services, we have seen transaction processing systems successfully used for coordination of inter-organisational work in precisely this manner. For example, consider the purchasing of a home entertainment system example shown in Figure 1. The on-line shop interacts with its specific suppliers, each of which resides in its own business domain. The work necessary to obtain each component is modelled as a separate task. In this example, the HiFi task is actually composed of two sub-tasks.

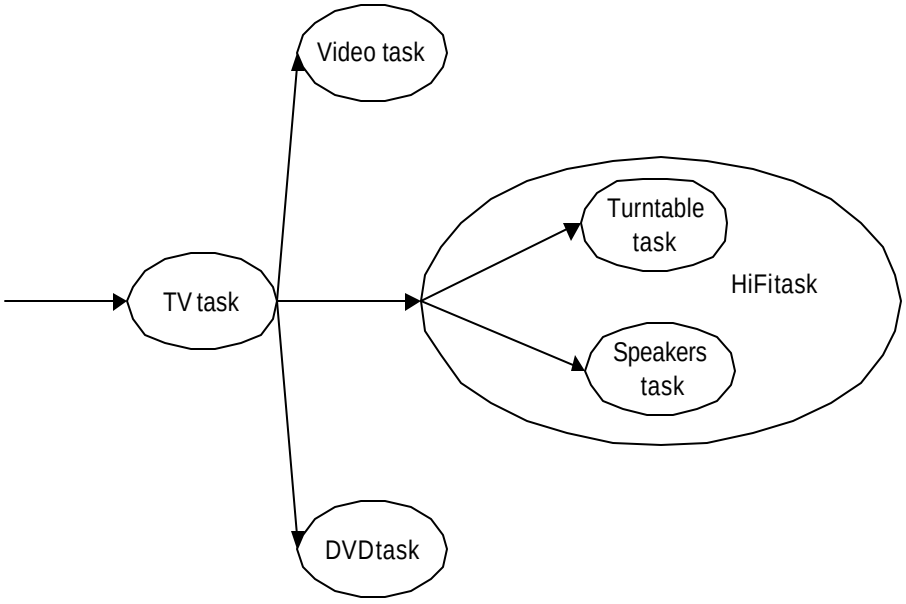

Figure 1: Business processes and tasks.

In this example, the user may interact synchronously with the shop to build up the entertainment system. Alternatively, the user may submit an order (possibly with a list of alternate requirements) to the shop which will eventually call back when it has been filled; likewise, the shop then submits orders to each supplier, requiring them to call back when each component is available (or is known to be unavailable).

For example, Figure 2 shows how the home entertainment system would be federated into interposed coordinator domains. Each domain is represented by a subordinate coordinator that masks the internal business process infrastructure from its parent (e.g., workflow system). Not only does the interposed domain require the use of a different context when communicating with services within the domain (the coordinator endpoint is different), but each domain may use different protocols to those outside of the domain: the subordinate coordinator may then act as a translator from protocols outside the domain to protocols used within the domain.

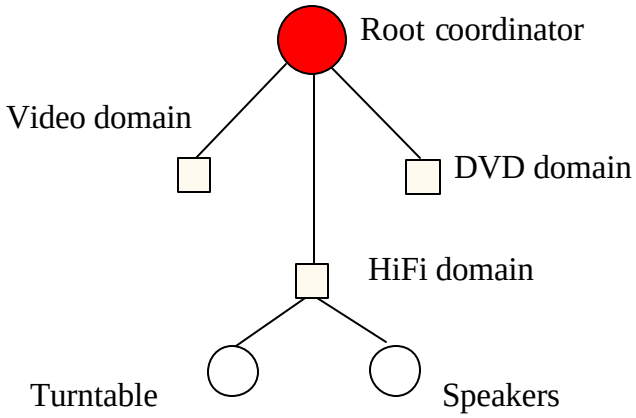

Figure 2: Example business process interposition.

Web services make the coupling of disparate business domains easier by breaking down interoperability barriers. However, they also bring to the foreground the coordination of inter-organisational transactions and how this can be achieved when each organisation has previously invested in non-interoperable infrastructures. As we have outlined, the solution to this is not to re-invent reliable coordination but to use what already exists.

It has taken decades to evolve transaction processing systems to their current levels of reliability, efficiency and performance. Even with the benefits of 20-20 hindsight, it is likely to take many years of effort to re-invent the wheel and assure users that what they are being asked to rely upon has the same level of pedigree as something they know they can rely on because of past experience.

References

[1] Business Process Execution Language for Web Services, http://www.ibm.com/developerworks/library/ws-bpel/, August 2002.

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

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