

Solving the mediation challenge: The Heart of an ESB

Introduction

There is growing acceptance that the adoption of Service-Oriented Architectures (SOAs) can offer significant benefits, specifically in terms of reduced costs and superior business agility.

This document looks in detail at the central challenge associated with building applications within an SOA – the process of ‘mediation’ that enables multiple and diverse services to work together. Previous approaches to mediation have been costly, complex and undermined the business rationale for enterprise integration projects.

A new approach – Progressive Element Refinement – enables mediation to be configured, not coded, and so secures the benefits associated with the adoption of SOA.

More on mediation

Mediation refers to the set of tasks that must be performed on each message or document so that multiple services (in the broadest sense of the word) can be combined to automate business processes spanning multiple applications, departments and even organizations.

Although some may regard mediation as simply the most recent addition to the SOA and Enterprise Service Bus jargon, its emergence is in fact hugely significant: it is now apparent that mediation is **the fundamental issue** that any viable ESB product or solution must address.

What exactly IS mediation? To make the concept more concrete, examples of mediation steps associated with a simple foreign exchange contract in financial services might be:

- Validating the message against that company standard
- Mapping of the customer name into the correct customer id format (by looking up a database) required by the FX order system
- Transforming the incoming message order format into the format expected by an FX order system – probably filtering out data not required by this system but required by audit or risk systems.
- Ensuring the sender of the order is allowed to order that value of transaction.
- Extracting the value and originator of the order and updating the financial risk engine.
- Enforcing management controls and service level agreements that might restrict particular quantities and currency being ordered.

In other words, within even a single step of a business process there will be many mediation steps that need to be completed; each acting on different parts of the message in different ways.

Although the term 'mediation' is new, it should be obvious that we are simply recognizing and giving a name to a challenge that has always been with us. Until recently hand coding was the only way in which most mediation tasks could be addressed, but now other options are becoming available. And how these mediation steps are coded or configured can mean the difference between success and failure when delivering integration solutions.

To take an example, without incorporating comprehensive mediation capabilities, the ESB is little more than "messaging plus" - a framework requiring extensive code customization to solve any but the most straightforward integration problem. Quantifying this coding effort, BEA's director of Product Marketing, Bill Roth, recently stated that integration projects are typically 75% code – with the bulk of that code probably being mediation related. Every line of code increases costs, but more importantly it reduces the flexibility of the completed solution - undermining the fundamental RoI argument for adopting ESBs and SOA.

Why do we need mediation?

Mediation is required because differences in data models, service definitions and the granularity of software services, makes communication between applications much more complex than might be imagined at first glance. The oft-proposed 'solution' to these issues - the act of creating a 'standards-based' service - only goes part of the way towards delivering truly flexible, loosely-coupled integration architectures (which in turn deliver business agility - let's keep this front-of-mind, it is the whole point after all).

This may seem to be heresy to some proponents of SOA, who believe that the perfect service definition should never need mediation, or alternatively that the solution to any mismatch is to create a new service definition. Based on that belief, some propose that service orchestration (usually based on the BPEL standard) alongside the 'correct' service definition on their own will solve any integration requirement.

To be blunt: this is optimistic and naïve nonsense. The business environment is more dynamic than ever before, and the rise of the 'virtual enterprise' will see the automation of business processes across multiple organizations. Even if it were technically possible to create 'perfect' service definitions and data models, no single organization will be able to mandate such a definition across new and existing technologies in multiple partners, clients or suppliers. Similarly, the alternative of re-designing a service definition to suit each new format requirement is equally unacceptable when dealing with mission critical systems: it is too disruptive and over time the number of service definitions will become unmanageable.

Even the proponents of BPEL as a complete integration standard recognize this and have created BPEL-J, which attempts to provide mediation by injecting fragments of code, which act as mediation band-aids, into the BPEL definition. Unfortunately, as the complexity of the problem increases, this approach becomes unbalanced and BPEL-J extensions begin to overwhelm the BPEL itself. Fancy jargon aside, that is a code-based approach, and while you can use a band-aid for a minor cut, you wouldn't rely on it for major heart surgery - which is the medical equivalent of enterprise application integration problems.

Implementing mediation is hard

The scale and complexity of mediation is something that has been realized – the hard way - over time. It revolves around the data or payload, and many mediation steps can be required, with each step consisting of a transformation, enrichment, or validation of the data – as well as splitting and combining of multiple messages. Managing and co-ordinating these steps in order to automate complex business processes is a non-trivial undertaking – and a host of integration approaches, including EAI, have ultimately failed due to an inability to manage complex sets of mediation steps in an efficient and effective manner and resorted to coding their way out of the problem.

These failures have only confirmed the importance of mediation. And they have demonstrated that failing to take the challenge of mediation seriously leads to excessive code, a lack of flexibility, and project failure.

On that basis, it is fair to conclude that mediation is **the fundamental issue** that any viable ESB product or solution must address. Without doing so we are simply replicating the mistakes of EAI. Specifically, an approach is required that performs mediation without code, and thus preserves the 'loosely coupled' architecture that in turn delivers business agility.

That approach is something PolarLake calls Progressive Element Refinement (PER).

Progressive Element Refinement – The Right Way To Address The Mediation Challenge

Mediation can be handled in a number of ways, and by approaching this issue in the wrong way some of the integration mistakes of the past are liable to be repeated. With this in mind, lets look in more detail at mediation, and why Progressive Element Refinement is absolutely central to success in this area. [It should be clear that 'element' in this context implies the XML element level, which can be a complete document, a sub-tree or a single data item.]

BPEL + XSLT ≠ Mediation

Before we do that let's remind ourselves of the present state of play for mediation in the Web Services/XML integration space. Until now most vendors have tended to associate mediation with the very broad term 'transformation', and then take a huge leap to assume that XSLT is therefore the appropriate standard for mediation of any document or message (the term document is used interchangeably with message when describing an XML document containing business data). Lets be merciful and pass over the well-known issues associated with XSLT - complexity, poor performance, the need to add extensive hand-coded extensions to fill in functional gaps and the inability to handle the common requirement for many-to-many transformation - and focus on the more fundamental point: *The simple technical transformation that XSLT supports is not mediation* - mediation is much more complex, and it requires multiple transformations, enrichments, message combinations and so on.

Unfortunately this means that even when XSLT is used as part of the solution, in most real-world cases significant hand coding is still required. As a result we are effectively hard-wiring these connections, undermining the benefits of a loosely-coupled approach, and failing to deliver the business agility that, as we noted above, is the whole purpose behind the move toward SOA.

Looking at the problem in this way, the challenge is more clearly defined: delivering mediation in a way that also maintains loose coupling. Any code increases the 'tightness' of coupling between services, and hence the goal must be to reduce the need for code as much as possible.

Slowly, some vendors are beginning to come around to this way of thinking, and we are beginning to see pitches for "code free" integration, typically combining BPEL (for orchestration) with XSLT-based transformation. This approach does of course work in simple environments characterised by relatively small messages and straightforward business processes, typical of the worked examples necessarily used in demonstrations and manuals. In the real world of complexity - which is, after all, where all of us must earn our living - these approaches suffer from the combined limitations of BPEL and XSLT, and the developer must revert back to code to fill the gap.

Progressive Element Refinement

Within even a single step of a business process, there will be many many mediation steps that need to be completed; each acting on different parts of the message in different ways. As we will see, Progressive Element Refinement addresses each component of this problem by:

- Allowing the granularity of the processing to be set at the element level – it enables the relevant part (and **only** the relevant part) of the message to be accessed directly for processing,
- Passing each element through potentially many processing steps, it supports the completion of the multiple complex sequences of mediation steps
- Providing an organizing structure to handle the context and control the complete mediation process.

This ability to handle complex processing requirements within a controlling context is the reason why PER can work and deliver code-free mediation.

The first difference between this approach and existing alternative approaches (including BPEL and XSLT transformation) is as simple as this: These approaches attempt to perform the common mediation functions (transformation, enrichment, routing, etc) on the **whole document** only.

Why does that matter? Well, there are a number of reasons, but two in particular stand out:

- Documents tend to be large – often very large – and processing whole documents presents serious performance issues. This is one of the reasons why XML was slow to be adopted within enterprise integration projects where performance really mattered, such as the delivery of STP projects in financial services environments. However, when operating at the element level, it is possible to stream arbitrarily large documents and process them as they arrive or leave.
- Working with whole documents makes the modelling and automation of complex business processes almost impossible. Only in a small minority of cases are simple 'one-to-one' transformations sufficient. More commonly, business processes rely on the repeated splitting, merging and recombining of data from multiple documents, all the while performing the common mediation tasks (enrichment, transformation, validation, routing and handling exceptions) upon only relevant data elements rather than the entire document.

Both points can be addressed by allowing the granularity at which the information is processed to match the task at hand. In other words, moving to the element level - clarifying again that 'element' in this context implies the XML element level, which can be a complete document, a sub-tree or a single data item.

So, in order to meet the requirements associated with complex and high performance business processes, *mediation functions must be performed at the element level rather than the document level*. This should not be surprising: most integration steps require access to only part of the entire message (the value of a contract, the return address and so on), not the entire message as is assumed in the SGML-based document processing world from where XSLT and other standards emerged.

However, the move to element level is not sufficient on its own - if we simply broke up documents into their constituent elements, the number of fragments would become unwieldy. The context needs to be maintained. Elements must be isolated and modified whilst still associated with the document or even documents they are part of. The maintenance of the appropriate context is what enables 'progressive refinement': modifying individual elements through the many many mediation steps, and potentially iterating again and again through the modified elements, until the mediation tasks are completed.

A secondary advantage of splitting up documents into typically smaller elements is that it allows a final switch away from code and towards configuration. Although coding within a PER approach is certainly better than coding at the document level, any code still tightens the coupling and tends over time to become complex and hard to maintain. Furthermore, introducing code forces the XML-based data to be translated into and then out of yet another (Java or C# programming model driven) data model. A side effect of splitting up documents into typically much smaller elements is that it has enabled a standard set of mediation components to be used in order to manage the complete task of mediation *without code* - and without the loss in performance that 'code-free' solutions sometimes deliver.

By taking the Progressive Element Refinement approach in its XML Data Circuits, PolarLake provides through *configuration* rather than code the entire process of:

- Isolating relevant data elements
- Defining the many mediation functions that will act upon them (and the potentially iterative sequence in which they will do so)
- Recombining these data elements into new documents.

All within a structure that can maintain the context of the entire process.

And this brings us back to deliver a prime benefit of SOA: When business requirements change, the configurations (which PolarLake calls circuits) defined can change with them without extensive reworking.

Acting on the whole document, on the other hand, it would never be possible to take this approach –standard mediation components could never handle the complexities of entire documents: the set would have to be too large. And if this approach cannot deliver, then development teams will fall back on the coding approach in order to manage element-level mediations – with a corresponding loss of business agility.

Conclusions

It has become increasingly accepted that Mediation is a key attribute of any ESB and in fact any SOA implementation. In fact, it could be argued that mediation is comfortably the most important attribute, and failure to address it results in excessive code, a lack of flexibility, and heightened risk of project failure. How important an attribute: It probably represents 75%+ of the total project cost – just as BEA says.

What does this mean for organizations considering SOA or ESB adoption? Most importantly, it suggests that they should be on their guard when selecting any product that claims to implement SOA. If mediation isn't adequately addressed, the benefits that are, ultimately, the whole point of SOA adoption, will be undermined and over time lost as the mountain of code increases. If it is addressed, and in my view that means taking a Progressive Element Refinement approach, mediation can be delivered in a way that supports the code-free automation of even the most complex business processes.