A practical approach to implementing a Service Oriented Architecture

1 Introduction

How do you balance the need to achieve an early success with SOA against the requirement for an architecture that will deliver long term success? You don't want to get bogged down in architectural committees for three years, but you don't want to make short term decisions that will be roadblocks to long tem success.

If your first project is to deploy a small number of reusable services in a matter of months but you know that four years from now you might have hundreds of services, an enterprise SOA infrastructure and specialized governance technology you have a dilemma. How do you move forward quickly without defeating your ultimate strategy?

One practical solution is to use a single enterprise integration platform that incorporates all of the functionality of an ESB and SOA backplane and can scale to the largest solutions. By using a single development and management technology Ensemble you can move forward quickly so that your first projects are successful in record time and ensure that in the long term, you have a scalable robust solution.

2 Components of an SOA solution

SOA requires more than a set of Web Services and a UDDI server. Even if all applications and services were exposed as SOAP, there would still be the need for semantic transformation, orchestration, business process management and governance. In the real world of legacy applications, databases and proprietary or domain specific protocols even more is needed.

This functionality is often provided by multiple products in a complex, distributed architecture but can be delivered more effectively and efficiently by a unified integration architecture.

2.1 SOAP Connectivity

The most basic functionality required in an SOA is to consume and expose Web services, and the simplicity is critical to the success of the project. But here you have another dilemma. There are many products that make it easy to use web services but don't give the power to support the myriad of WS standards that will be important down the road. But many platforms that provide that functionality are often little better than starting with a java compiler or .NET development environment.

Ensemble provides the ability to very easily import a WSDL , consume a web service, and to project legacy functionality as web services. However it also provides more sophisticated capabilities for WS standards, customized SOAP headers and more, without significant added complexity.

2.2 Connectivity with non-SOAP applications

Where the environment includes non-SOAP applications or services, adapters or connectors are required for these protocols.

One approach is to write local wrappers around all services to expose the functionality as a web service and to write plug-ins for all the applications so they can consume web services. While this can be effective and minimizes the complexity of the transport across the network, it places a high demand on the groups responsible for maintaining legacy services to deliver in accordance with latest

A second approach is to implement distributed agents that run local to the legacy applications and communicate with the legacy services in their native protocols and communicate across the network in a common protocol. This places less load on the legacy teams and can also be effective. The downside of this approach is the management, recovery and maintenance of distributed agents. It also makes it hard to adapt to custom protocols and the adoption of adoption of specialized adapters such as screen scrapers, which are sometimes unavoidable.

The third approach is to allow connectivity over the network from the integration hub over the various legacy services native protocols. This requires a central platform with a wide range of adapters that is capable of high throughput, and

can be distributed for very large enterprises. This approach imposes the minimal load on the support teams for legacy applications and avoids the complexity of installing and managing distributed agents. This approach also gets to the first success

2.3 Mediation and Transformation

Protocol mediation and Semantic transformations are often confused or combined. Protocol mediation addresses the need to convert between protocols such as JDBC, SOAP, flat files or EDI transports. In a pure SOAP environment this may not be necessary but in most environments it is.

Semantic transformations take account of different format at the application data. This may be as simple as the format of a date, or it might be product codes that have to be translated from tables, or a complete restructuring as for example to convert an 'invoice' between the structures used by two different applications.

2.4 Service Discovery

Design time discovery of services, together with details of the interface and service requirements are critical elements for success of a large SOA implementation and UDDI V3 is standard approach with many low cost server implementations available.

One hospital who had recently started developing a SOA, told me that they have plans to implement a UDDI server and having that as a guiding principal early on helped them make some good design decisions, but with a small number of services implemented they didn't need it yet. At this stage the SOA development team were actively campaigning for people to use their services – 'Discovery' would come later when the organization had got used to the idea of using services.

For fully established SOA implementations, the UDDI repository will hold not only definition of the service interfaces but it will contain more than just a WSDL. It also provides information about SLA, and other information about how the service should be used

When choosing an infrastructure technology the ability to work with UDDI as Ensemble does, is important and it is essential in your early thinking to understand that is where you are going, but like the hospital described here it is

not necessary to implement the UDDI server before going live with your first service.

2.5 Governance

Ensuring the service levels, security and strategic direction of your service oriented architecture meets the policies laid out by your organization is a difficult and wide ranging challenge. Many of the necessary steps are not technical, such as defining funding models that promote development and use of reusable services. There are however, important technical features are required for good governance of your SOA implementation.

Information on the numbers of calls to each service, response times and availability are needed to assess whether the system is meeting service level agreements to plan resources and to respond appropriately if service levels are deteriorating.

An understanding of who is using the service network is required for a number of reasons. A prime reason is the security audit trail to detect misuse of the system is clearly essential, but information about legitimate use is also required to enable optimization of the architecture, appropriate charge back possibilities, funding justification and assessment of the return of investment in the architecture.

The Ensemble message store, provides information about all traffic passing through Ensemble including details of response times, availability and error reports. The Ensemble audit database provides additional information to identify who is using the system and any additional security events.

3 Business Process Management

Often discussion of the SOA is restricted to the discovery and communications between systems, possibly with an ESB to handle protocol mediation. But by adding the workflow and Business Process Management to the architecture, immediate benefits are derived.

BPM brings the dual advantages of orchestration of services at the integration layer to expose 'composite services' and the support of long running business processes.

Ensemble provides a graphical way to define orchestrations of back end services. The orchestration of calls to external functionality, combined with business logic

and data transformations within Ensemble allow you to expose complex business processes as a service or as part of a composite application. Because Ensemble can directly communicate with databases and non-SOAP applications, these business processes can very quickly add value to your business.

Because business processes often involve requests that don' get an immediate response business processes have to be long running and survive system restarts. For example if a business process that handles the acceptance, and fulfillment of an order from inventory can't be satisfied because a part if temporarily out of stock, the business process might issue a re-order request and wait (indefinitely or with a prescribed timeout) for an event that indicates the part is now available.

Workflow solutions involving people addressing certain actions couple with orchestration of computer systems is a classic example of long running business processes.

4 EDA in SOA

Discussion of BPM and long running business processes leads directly to the larger topic of Event Driven Architecture and SOA.

While Ensemble's support of 'Complex Event Processing' is outside the scope of this paper, simple events of various types are important to anything but the simplest of SOA solutions.

The simplest case is with simple messaging systems such as EDI or HL7 interface engines. These are essentially event driven although rarely regarded as such. The only reason it is significant is to highlight how often these messaging systems are completely separate from the SOA solutions. It is actually Ensemble's ability to support both EDA and SOA in the same architecture that allows it to seamlessly move incorporate services into a messaging engine and vice versa.

Event drive architectures are based on the detection and receipt of events. Ensemble can detect or receive events using a variety of inbound adapters. These may be listing for a message over various protocols such as MQ, HL7, X12 or JMS and making a call to an external service or invoking a business process that orchestrates calls to many external services, databases or applications. Ensemble will also detect events by examining external systems such as polling external databases or looking for files being created.

The ability to detect events and receive event notifications is critical to broadening the narrow view of SOA as a 'web services only' architecture and address the practical integration issues of a typical enterprise.

5 Conclusions

It is possible to deliver results from a Service Oriented Architecture quickly without compromising the long term goals.

The complete integration of an enterprise requires SOA, EDA, BPM and other integration capabilities.

An SOA plan that assumes SOAP will run into many obstacles.

SOA and EDA are partners in success. In the real world, one complements the other and any integration platform you choose must support both paradigms.

Contact InterSystems at … and ask how Ensemble can solve your integration needs.