The Role of SOA in Enterprise Data Integration

Jonathan Carter
Enterprise Architecture Solutions

• What role does SOA have in Enterprise Data Integration?
Introduction

- The Role of SOA
- Case Study
- Lessons learned
- Future

- By looking at a real-world example – of Exchange Rates data – we’re going to show how SOA can change the way in which we do data integration.
- From that, we will go on to demonstrate what role Service Oriented Architecture can play in Enterprise Data Integration
- We will look at drivers and issues behind this particular scenario
- We will then go on to describe the strategy that we devised - describing why and how we introduced SOA
- After that, we will see how the solution architecture for the scenario looks after the roll out of the strategy.
- Then we’ll take a step back and look at the lessons that were learned – in particular concentrating on the things that you need to take care with when implementing
- We will review our plans for the future of this implementation
The Role of SOA

- Allows you to evolve the data integration and information architectures
- Improves speed-of-delivery of solutions
- Ubiquity of Web Services means the technology barriers have been brought down
- Abstracted view of the data
- SOA, not SOT
  - Technology constraints
  - Currently, not the answer for everything
- Strength lies in providing access to common data

What is the role of SOA?
- As a preview of our conclusions, here is the role.
- SOA enables you to evolve data integration and information architectures incrementally
- SOA improves the speed of delivery
  - No coding to add consumers,
  - No infrastructure to add for each new service
- Web Services can deliver ubiquity for SOA
  - The tools and package vendor support for Web Services is now very good and widespread
- SOA provides an abstracted view
  - Which delivers flexibility for data architecture strategies
- This is Service Oriented Architecture, not Service Oriented Technology
  - SOA is technology independent
    - It’s a concept or approach and a set of principles
  - SOA can support Data Integration requirements but there are some technology constraints.
- Currently, SOA’s strength lies in providing access to common or shared data
  - Reference data
  - Master data, etc.
- Don’t confuse doing Web Services with doing SOA
- SOA is about a
  - Shared
  - Controlled
  - Well-defined set of interfaces that are
  - Managed
- Background to scenario
- Global FMCG – 180 operating companies
- [CLICK] Sharing data around the world between companies
- [CLICK] Globally deployed EAI solution in place
- 80% integration is data
  - Important that SOA was applicable to data integration
- [CLICK] New IT strategy
  - This had Implications for the integration approach in general going forward
Scenario

- Exchange Rates data
- Distributed to many systems
  - across the world
  - including several SAP systems
- Updated daily
- Straight-forward data set

- Focus on one of the integrations – the Exchange Rates data
- A refresh of the integration technology was already underway, so work was already planned for this solution.
- The data is distributed around the world to a large number of systems
  - Several SAP
  - Oracle
  - Lotus Domino etc.
- The data is updated daily – normally
  - This introduced some issues later on to do with bank holidays, weekends etc. when the data was not updated every day.
- The structure and content of the data set was not too complex
- All these things made this an ideal candidate for a pilot solution.
- This is the “before” picture - an integration solution built using Message Oriented Middleware.
- [CLICK] The source data is pulled from a Financial content provider using FTP and loaded into a data warehouse
- [CLICK] Extracts are taken as XML messages
  - and sent to the message broker which forwards them on to registered recipients
  - this registration is coded into the message broker
- [CLICK] However, there is also some “back door” integration going on
  - SAP – SAP (as the SAP teams find this easier)
  - and also access straight into data warehouse
- There are good reasons for this “back door” integration, though.
  - Although there was the global EAI solution
  - The cost, the time and the skills required to use it were too large
- And this made the time and cost of integration too much for some solutions
- Because of these hurdles to the use of the EAI solution, we call this a “High Impedance” architecture
Data Integration issues

- Multiple integration technologies
- Multiple access points, no “single truth” for data
- Difficult to manage load on data sources
- Multiple security models and mechanisms
- Unclear who owns the integration solution
- Inconsistent mechanism for meta data sharing
- Data usage tracking
- Inconsistent data quality

- A review of the existing solution exposed the following (possibly familiar) issues for data integration
  - Multiple integration technologies to share data
    - And with multiple access points and then multiple data hierarchies
    - This makes it very hard to track relationships between consumers and providers of data and the various copies of data
    - This makes it very hard to manage or understand the impact of change
  - It is difficult to manage the resource load on the data sources
    - with multiple clients and direct access to the data sources
    - This exposes the source data too openly and makes it hard to manage the resulting load on the provider systems
  - Multiple security models and security mechanisms are required to serve the multiple clients with their multiple access technologies.
    - The Quality of Service and interaction style is dictated by clients’ access technologies.
    - Which means that the provider has to be able to support each client’s technology and security model.
    - Which in turn makes it very difficult to manage security holistically over all these models and technologies.
  - It’s unclear who owns the integration solution. Who do I got to for if I need a change?
  - There’s an Inconsistent mechanism for meta data sharing.
    - There’s no central meta data repository
    - The semantics of the data become inconsistent
    - New formats for what is semantically the same data are created and must be maintained
  - It is hard to track usage of the data and therefore hard to understand the impacts of change
  - There is Inconsistent data quality because of the various formats and copies of the data. This makes it very hard to understand
    - Who has latest copy?
    - Who’s using the latest copy?
    - What’s the correct value for data, now?
Strategic drivers

- New IT Strategy
- Convergence and consolidation of technology
- Cost reduction
- Simplification and clarity
  - Infrastructure
  - Organisation

All of these had implications for the integration architecture

- In addition to familiar issues above,
- There was a new IT Strategy
- The drivers of this were:
  - Convergence and consolidation of technology
    - To reduce the technology portfolio
    - To reduce the number of technology deployment locations
  - Like most organisations to reduce costs
  - To simplify and clarify the infrastructure and support organisation.
- All of these drivers had implications for the Integration Architecture in general
- As we mentioned earlier, 80% of the integration in the organisation is data integration
  - So, we had to define a strategy for the data integration architecture
  - Whilst ensuring that we did not create later problems for the
    - Application Integration Architecture
    - Process Integration Architecture
- We had to define a Strategy & Architecture for Data Integration
- We followed our standard EAS process
- We defined a vision that was based on SOA
  - Because SOA is about minimising impedance in the architecture
  - Which means that it makes the interaction between data providers and consumers easier
- We defined the required strategic technologies
  - Remember, SOA is a concept and set of principles and we need implementation technology
  - We selected Web Services for implementing SOA
- To introduce SOA and the technology in a controlled manner, we defined a phased Roadmap
- The Phase 1 pilot consisted of
  - Identifying candidate services
  - With an internal scope
- The Phase 1 rollout
  - Was delivered as an Enterprise capability, not a point solution
- Let’s take a more detailed look at this process
- SOA is basis of vision
- You may well be familiar with the concept of a Service. However, especially with the explosion of Web Service technology, there are many - often conflicting - views of what a service is.
- Here is our definition
- What is a service?
  - It must have some utility and be delivered in a discrete package
  - It must be designed to be shared – which means it must not contain any consumer-specific logic
  - It must have ubiquitous access so that it can be used by anyone, anywhere
  - It must be self describing in terms of what it does and how it is to be used
    - This is achieved through a well defined interface
- The purpose of SOA is to separate the what from the how
- I don’t want to worry about how it works – as long as what it does is consistent with the specification in the interface
- This approach and these principles provide an abstraction layer
  - That provides a controlled evolution of the architecture
  - Enabling us to pick right technology for a solution and even defer major technology investments
  - Which means that we can build the architecture incrementally
- Logically, this is how we wanted our services to work
- And how do we develop new consumers of the service?
- From our scenario, we have an Exchange Rates Service providing data that is stored in a Data Warehouse
  - [CLICK] The service has a defined interface or contract
    - Once this interface has been implemented
    - We load service interface into the service registry
    - Consumers can then start using it.
  - [CLICK] For example, a Procurement application developer needs to use the service
    - They retrieve the service interface from the registry
    - [CLICK] Then at runtime, the application uses service
  - We kept the concept simple to start with
    - There are no dynamic invocations
    - We will get enough benefit to justify the approach even by pulling the interface from the registry at development time only.
    - Additionally, consumers are responsible for any data transformations that they require from the schema that the service returns.
      - We are not building a system-wide “Esperanto”
      - However, we are semantically providing the data that is required by systems
  - The service provides us the abstraction enabling us to
    - Change the database of the data source
    - Change underlying data format of the data source
    - Change content provider, e.g. the content provider could provide the source data by a Web Service.
  - This concept seems straightforward
    - But as the number of services and consumers grows, this model rapidly becomes complex
- So we provide a service infrastructure to manage the complexity, and bring order to the chaos of the sea of services

- [CLICK] At development time, there is a single place to get meta data about all the services
  - Which describes what the service is, how it is invoked, where it is invoked, the Quality of Service that it provides, the credentials that must be supplied in order to use the service etc.
  - However, the details of these may vary consumer to consumer
  - So, we took a policy-based approach to manage this with the policies in the service registry
  - [CLICK] The Policies are defined against roles and the consumers are assigned a role

- [CLICK] At runtime the policies are applied by the service infrastructure

- [CLICK] The Service Infrastructure provides
  - Failover, load balancing, accurate monitoring (and detailed logging)
  - Security and pro-active alerting, e.g. to spot any service endpoint failures before consumers attempt to use the service

- The policy configuration is also used for
  - Version control - since things will change
  - Access control not only authorisation but role-based access. E.g. To ensure that if a developer requests a service then they are routed to an implementation of the service that is in a development environment rather than production.

- [CLICK] The Service Infrastructure provides the Enterprise Class capabilities that we need for SOA.
  - The policy-based approach makes it manageable as the numbers of services and consumers scale up
  - The service infrastructure also separates the consumers and the providers
    - Physically, so that the consumers do not need to know the details of where the service physically resides
    - Technically, so that consumers can be developed using the technology most suitable for the consuming applications and service implementations can be developed using the technology most applicable to the nature of the data source.
- To be able to rollout this concept globally, we need to ensure the service infrastructure is not a single point of failure [CLICK]
- Through the Service network we get the ubiquity to consume services in a globally scalable environment
- We want to have a single logical registry [CLICK]
- But with multiple access points to the infrastructure
  - For scalability of the solution
  - Combined with load and network performance management.
- [CLICK] Now we can have local access to the global infrastructure
- While the Service Network abstracts the physical endpoints, allowing us e.g. to move them, replace them or redeploy them.
- Making a request in this concept is like posting a letter. If I want to send a letter to someone, I post it into my nearest local post box and – as long as I have correctly addressed it – the letter is delivered by the postal infrastructure within my expected timeframe. I don’t care how it gets there or what route it takes, as long as it gets there.
That is our vision for how we wanted the SOA to work and what we wanted to deliver.

So, we need an implementation technology for this SOA

People have been implementing SOAs for many years
- In EAS, we have experience of using CORBA in the late 1990s and then later using Java RMI
- However, like most people we had some problems achieving all the capabilities of SOA, particularly the ubiquity

However, Web Services seems to have addressed many – if not all - of these issues

[CLICK] So, in our strategy, we use WSDL to describe services and SOAP to invoke them

[CLICK] Inside the service infrastructure we continue with the WS-standards

[CLICK] Those are our strategic standards. In terms of technology products to deliver these,

[CLICK] The SOA management tools (the registry and service infrastructure) are provided by Blue Titan

[CLICK] For service implementation we used Cast Irons Systems Application Routers, since, in this organisation, this was already strategic integration technology. However, we could have used any other Web Service-capable integration technology, E.g. If the data source was an SAP system, we might have used SAP’s Netweaver
SOA strategy addresses the issues

- Multiple integration technologies
  - Common and robust technology infrastructure with ubiquitous access
- Multiple access points, no “single truth” for data
  - Provides abstraction layer for evolving your integration architecture in a controlled manner
- Difficult to manage load on data sources
  - Abstraction protects data source
- Multiple security models and mechanisms
  - Manageability through policy and role-based approach
- Unclear who owns the integration solution
  - Devolves organisation integration responsibilities
- Inconsistent mechanism for meta data sharing
  - Inherently centralises meta-data
- Data usage tracking
  - All access provided through service interface
- Inconsistent data quality
  - Single interface to common data

- So, how does our strategy address the issues that we were facing at the start of this scenario?
- We had multiple integration technologies, which our SOA strategy addresses by providing a common and robust technology infrastructure with ubiquitous access.
  - The development tools support for Web Services now makes it easier for consumers of the data to work in right/strategic way, rather than opting for diverse solutions that are “easier for them” [CLICK]
- Multiple access points a no “single truth” of the data are addressed by the abstraction layer that enables us to evolve the integration architecture in a controlled manner.
  - We can implement an ordered evolution of integration technology
  - Manage data that is in fact duplicate [CLICK]
- It was difficult to manage the load on the data sources and SOA resolves this by the abstraction layer that protects the data source.
  - The real endpoints that provide the data are obfuscated – you can’t see where they are or who they are.
  - The capacity to handle requests is managed away from the data source, by the Service Infrastructure.
  - [CLICK] Enabling us to balance the load on and remove the direct access to the data sources
- The policies and roles approach manages the multiple security models and security mechanisms
  - We can now manage the Quality of Service requirements of the consumers
  - The point-and-click configuration of the tools means that no coding is done in the services to manage the security requirements and simplifies the implementation.
  - We get all the benefits of centralised management in terms of a single place to monitor, manage and maintain policy
  - However, we have distributed enforcement of the policies, thereby avoiding potential performance bottlenecks or single failure points [CLICK]
- For many data integrations, it is unclear who owns the solution. Our SOA strategy devolves the responsibilities
  - Service implementers are responsible for the service
  - Consumers are responsible for the integrations that they have implemented in order to use the service.
  - Responsibility for these can change over time but now we have a central to registry go to for all the information we require about of the service.
[CLICK]
- That the SOA strategy inherently centralises meta data addresses the Inconsistent mechanisms for meta data sharing.
  - The meta data is centralised
  - There is a standard and common mechanism for sharing meta data
  - The Service Registry is the single place to go for meta data
  - [CLICK] Which provides consistency for sharing meta data
- Data usage tracking was very difficult by our strategy enables us to track the use of shared data through monitoring in the Service Infrastructure [CLICK]
- Inconsistent data quality is addressed by having a single place to access data. The single interface to common data reduces the issues associated with copying data around the organisation [CLICK]
- That's our SOA strategy and there is quite a lot to it.
- In line with our process, we defined a roadmap to deliver the strategy and introduce SOA
- This had to be pragmatic – in order to ensure a successful introduction we needed to make sure we could walk before we tried to run
  - This meant that we had to first establish the SOA approach in a controlled scope
- The first step, [CLICK] was to evolve systems integration to Web Services
  - We introduced simple Web Services
  - But these were backed by the Enterprise Class SOA infrastructure
- [CLICK] This would be evolved over time
  - To include more complex scenarios and additional capabilities
- It is important to note that all this is achieved in context of overall plan for when and how capabilities introduced
- Coming back to our Exchange Rates scenario, this is how the Phase 1 Pilot architecture looked.
- Data is still being pulled via FTP from the financial content provider
- However, now we’ve introduced the Exchange Rates Service that is providing data that is held in the data warehouse. Applications now request the data from the service via the Service Infrastructure.
- How did we define this service?
  - We defined the service TOP DOWN. The schemas that we used were aligned to Information Architecture strategy and we worked closely with the Information Architects to define the data schemas
  - [Click] We defined schemas and operations of the service and only THEN did we define the service interface – using WSDL. Here’s a snippet of the WSDL.
  - We registered this WSDL with the service infrastructure [CLICK] and then
  - Consuming applications could pull the WSDL from the registry and can make the necessary SOAP messages [CLICK], which look like this.
  - In line with the strategy, Cast Iron Application router orchestrations were used for building the service implementation and Blue Titan for the service infrastructure.
  - [CLICK] This scenario also manages to illustrates how the evolution of data integration & information architectures is enabled.
    - After the pilot had been completed, a Cast Iron solution replaced the FTP script for sourcing the data from the external content provider with no side effects to the service consumers.
  - The pilot ran successfully and so, following our process, the next step was to roll out the solution.
- However, rolling out the Technology was only part of the story
- We also had to put the support organisation in place
  - Both in terms of operational support
  - And in terms of governance of the services
- This organisation is based in Malaysia
- Although most infrastructure is based in Germany
  - The initial instance of the Service Registry and the access points
  - Along with most of the service implementation technology
  - And the data warehouse which is hosting the data source
- Cast Iron technology was already in place at both locations
- [CLICK] Consumers from Australia, [CLICK] Germany and UK now use the service
- [CLICK] Malaysia hosts additional Blue Titan and Cast Iron technology
  - For infrastructure development testing
  - And additional, regional capacity
- Our solution only has one access point initially
- The Service Network provides scalability (both up and down) without impacting existing users of the service
- [CLICK] So, in order to ensure this, the consuming applications all access infrastructure
  - even when they happen to reside in same data centre as service implementation itself!
- [CLICK] The service implementation is deployed near to the data source, for both performance and security reasons
- [CLICK] However, all of this is supported and governed to enterprise quality by the Support Organisation
Lessons Learned

- Enterprise initiative
- Organisation buy-in
- Enterprise class infrastructure
- Ownership
- Useful service
- Service definition
- Schema management
- Don’t be afraid to grow slowly
- Control your scope

- We learned a number of significant lessons during the implementation of this real-world scenario.
- Firstly, SOA is an enterprise initiative
  - It’s not about implementing point-solutions
  - And it’s important that everyone realises this
  - You need to think about SOA as an enterprise activity
- As such it involves the wider Enterprise, so you need organisation buy-in
  - From the development community
  - The business community
  - The operational community
  - And importantly from Senior IT management
- Enterprise-class infrastructure is needed
  - To provide the management of services
  - It’s very hard to retrospectively apply management to services that have already been implemented and are being used!
  - It’s important to treat service meta data same as other meta data
    - And ideally, host it in the same repository
  - However, this Enterprise-class infrastructure doesn’t have to be a very large investment
  - If it’s the approach and intent that is the important bit
  - And you can scale this up incrementally from a modest starting point.
- Ownership is very important
  - Organisation responsibilities must be well understood
    - The business must own the semantics of the service, must as they should own the data
    - However, you can have a central IT team to implement and support the services
- The service has to be useful if you are going to demonstrate any value from this
  - The data provided by the service must be done so without compromising service integrity
  - The data must be shareable
  - To demonstrate the utility of the service in our scenario, as well as being shared by multiple applications, it supports more than one style of data integration style – ETL and what we call Remote Data Access (the online access by remote applications to specific items of data in the data set)
- A key point is that how services are defined is very important
  - They must be defined from analysis of the requirements
  - The service schema must be aligned to meta data strategy
  - This service must be defined Top-Down – and not by exposing the underlying database tables as web services
  - It’s the services and their operations that are important, not the underlying data schemas of the data store
  - Resist consumer-specific requirements
    - E.g. the service should not be delivering specific data formats for particular applications
    - There can be no consumer-specific logic in the operations, otherwise the service can’t be shared
    - OR perhaps worse, you end up with as many operations as you have consumers. Rather, provide operations that can be used by all consumers as shareable tools that can be used/combined to satisfy the consumer’s requirements.
  - For example, in our scenario, sometimes Exchange Rates data was missing for some days due to...
Benefits realised

- Increased speed of delivery
- Improved quality
- Higher level of development
- Usage controlled through configuration rather than coding
- Enterprise-wide monitoring, alerting and usage tracking
- ‘Point-to-point’ development, ‘brokered’ management and runtime

- In this small scenario, we realised a number of benefits
- Increased speed of delivery
  - The consumers of the service were impressed and very pleased with the speed of delivery of both the service and of the integration that they had to do in order to use the service.
- We saw an improvement in the quality of the integration solutions. In this case, we got it right first time!
  - Due to the well defined interfaces of the service
  - Having no consumer-specific logic in the service
  - And the Service Infrastructure allowed us to make enhancements added incrementally afterwards without impacting existing consumers.
- We have provided a higher level of development
  - We took the technology out of the development process for integration solutions.
  - Service implementers and consumer developers can concentrate on the important bits of their work rather than on coding technical middleware
- We controlled the infrastructure via configuration and not coding
  - Which means that we can add new consumers quickly without coding in the infrastructure or add new services with no additional infrastructure needing to be deployed.
- We have Enterprise-wide monitoring and alerting.
  - Which means that now we can track usage of the data, which was very hard before with the EAI and “back-door” integrations.
- We get all the simplicity of point-to-point development with all the benefits of a brokered solution at management and runtime
Extended capabilities
- Publish / subscribe
- Enhancements to security
- Meta data management
- Write-backs
- Transformation as a service?

Further SOA usage
- B2B data integrations
- Event-based data distribution

- Going forward we plan to extended the capabilities of our SOA infrastructure to include
- Publish / subscribe and expect to use the WS-Eventing standard for this
- Add security enhancements for B2B in particular
- Extend the service registry in line with the metadata management strategy
  - Our vision is of having the service registry as a logical part of the Enterprise metadata repository.
  - Having both services and data together
- Write backs.
  - Our scenario is about providing read-only access to data. Services that need to update the data in the underlying data store present some additional issues.
    - Writing data back to the data store reliably can be hard – especially when the service is storing data across multiple sources
    - We've been looking at EII (Enterprise Information Integration) technology for this capability.
- Data transformation as a service.
  - We stated earlier that the services would not perform any data transformations for particular consumers, we have been exploring the idea of providing specific services that only provide well-defined transformations between specific schemas. Consumers could then call such transformation services as required, e.g. to transform a response from another service into a format that the consumer can understand.
- For further use of SOA, we are looking at
  - B2B data integration
  - Event based data distribution where data is distributed to consumers in response to particular events.
The Role of SOA

- Allows you to evolve the data integration and information architectures
- Improves speed-of-delivery of solutions
- Ubiquity of Web Services means the technology barriers have been brought down
- Abstracted view of the data
- SOA, not SOT
  - Technology constraints
  - Currently, not the answer for everything
- Strength lies in providing access to common data

- From our experience of this real-world SOA scenario and implementation, we believe the role of SOA for data integration is
- SOA enables the controlled evolution of the data integration and information architectures
- Improves speed of delivery of data integration solutions
  - Through there being no coding required in order to add service consumers
  - And that no new infrastructure needs to be implemented for each new service that is developed
- The ubiquity of Web Services brings the technology barriers down, making it easier for consumers and providers to interact (significantly reducing the impedance of the architecture).
- An abstracted view of data allows flexibility of data architecture strategies, including
  - Technology migrations
  - Data source or data provider changes etc.
- This is Service Oriented ARCHITECTURE, not Service Oriented TECHNOLOGY
  - SOA is independent of technology – it’s an approach and a set of principles
  - It can support all of the requirements for Data Integration and we’ve used it for ETL, online access, dashboards etc
  - It is only constrained by the capabilities of the implementing technology
  - We used Web Services in this case, and we realise that the required XML processing could be a problem for some scenarios. However, we expect XML processing problems to be addressed by technology, e.g. dedicated XML processing appliances.
- Therefore, we believe that the strategy is valid
  - If you adhere to principles of SOA
  - You will retain the benefits
- At this point in time, SOA’s strength lies in providing access to shared, common data (e.g. Reference or Master data).
  - Integrations will continue to exist for things that are not shared.
  - And it’s probably not worth using SOA for these
  - You may want to use Web Services for them and that’s fine
- But don’t confuse using Web Services with having an Service Oriented Architecture
- SOA is about a
  - Shareable
  - Controlled
  - Well defined set of interfaces
  - That you can manage
- Adhere to the principles of SOA
  - And you can bring real benefits to Enterprise Data Integration
Questions?

jonathan.carter@e-asolutions.com
www.enterprise-architecture.com

If you have any questions or comments, we would be very interested to hear from you. Please feel free to contact us.