



Evolution of a Service-Oriented Architecture within the Wholesale Telecommunications Sector

Kerard Hogg, Programme Architect
Peter Chilcott, Senior IT Architect



Abstract

- **This presentation will outline the progress of a Wholesale Telecommunication provider who has grown their production internet facing Service-Oriented Architecture from its initial Web Services release in late 2003 to its current BPEL orchestrated ws-I compliant web services solution.**
- **The discussion will encompass business motivations, technology choices, pain points and further utilisation of the SOA stack.**



What is SOA? – As Defined by Wikipedia

- In **computing**, the term Service-Oriented Architecture (SOA) expresses a **software architectural concept** that defines the use of services to support the **requirements** of software users. In a SOA environment, **nodes** on a **network** **make** resources **available** to other participants in the network as **independent services** that the participants access in a standardized way. Most definitions of SOA identify the use of **Web services** in its implementation. However, **one can implement SOA using any service-based technology**.
- Unlike traditional **point-to-point** architectures, SOAs **comprise loosely coupled** (joined), **highly interoperable** application services. Because these services interoperate over different development technologies (such as **Java** and **.NET**), the software components become very reusable, e.g. the same **C# (C Sharp)** service may be used by a Java application, and/or any other programming language which can access this service, due to the virtue of the interface definition being defined in a standards-compliant manner (e.g. **WSDL**) which encapsulates/hides the vendor/language-specific implementation from the calling client/service.
- SOA **provides a methodology** and **framework** for **documenting enterprise capabilities** and can support integration and consolidation activities.
- High-level languages** such as **BPEL** or **WS-Coordination** take the service concept one step further by providing a method of defining and supporting workflows and business processes.
- Service = (Ideally) a self-contained, stateless business function** which accepts one or more requests and returns one or more responses through a well-defined, standard interface. Services can also perform discrete units of work such as editing and processing a transaction. Services should not depend on the state of other functions or processes. The technology used to provide the service, such as a programming language, does not form part of this definition.



What is SOA? – IBM Developer Works

- Service-Oriented Architecture (SOA) is a **component model** that **inter-relates** an application's different **functional units**, called **services**, through **well-defined interfaces** and contracts between these services. The interface is defined in a manner **independent** of the hardware platform, the operating system, and the programming language in which the service is implemented.
- These independent interface definitions are known as **loose coupling**. The benefit of a loosely-coupled system is its agility and ability to **survive evolutionary changes** in the internals of each service that make up the whole application. Tight-coupling means that the interfaces are tightly interrelated in function and form, thus making them brittle when any form of change is required.
- SOA is not new**, but is an alternative model to the more traditionally tightly-coupled object-oriented models. The SOA of today relies on the eXtensible Markup Language (XML) and a language called *Web Services Definition Language* (WSDL) providing flexible and dynamic interfaces. Other ways to implement SOA include CORBA and Message-Oriented Middleware systems eg. IBM MQ Series.
- SOA is an **architecture model**, you need more than just a service description. You **need to define** how the overall application performs its **workflow between services**.
- An SOA should relate the commercial processes of a business to their technical processes. Thus workflow also plays a significant role in the design of SOA.
- Workflow of a dynamic business can include operations with other external partners, which need policies of how relationships between services should transpire.
- Finally, must have **trust and reliability** to carry out the processes as expected according to agreed terms. So, security, trust, and reliable messaging should play a significant role in any SOA.

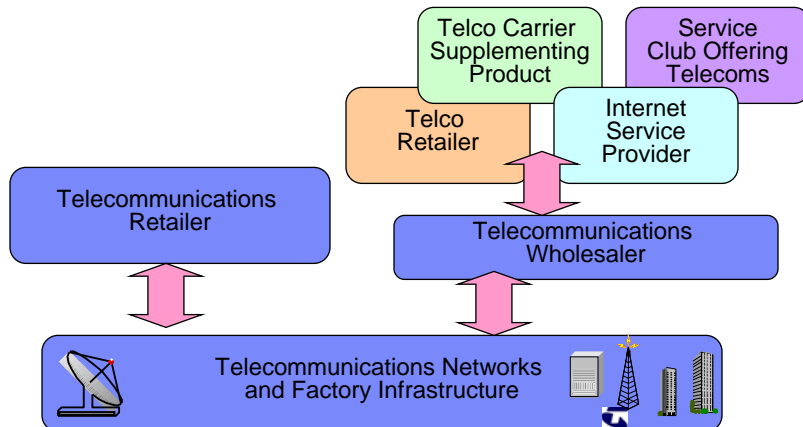


Agenda

- **Business Perspective**
 - What was the business scenario?
 - Solution – Historical Steps
- **Technical Perspective**
 - Technical Architecture
 - Web Services
 - Business / Application Services
 - Process Services

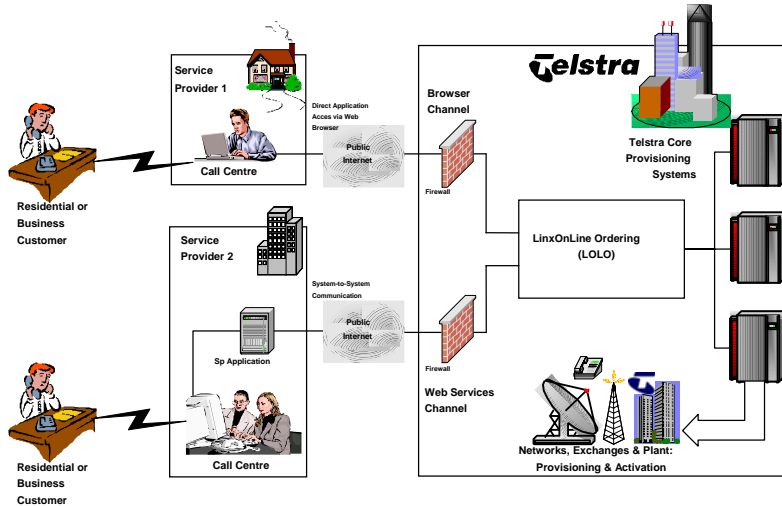


Business Context





Step Three - Solution Context



Key Architectural Observations

- **Agility and flexibility** come as much from the ability of the underlying services as from the process which might orchestrate them
- **Multiple layers of services** are required to service any reasonable size enterprise. **Granularity** will generally be finer further from the enterprise boundary. Hence each layer heading outward is aggregating those servicing it.
- **Conversational services** are, by their very nature, more tightly coupled than non conversational. And, we hypothesise, more likely to occur with human interaction.



BPEL IDE palette example

Choreograph
Process

1. Palette

– Contains activity icons which can be dragged-and-dropped on the canvas

2. Variables Area

– Contains process variables.

3. Activities Area

– The area process where the process is assembled

4. Action-bar

– Contains actions that are relevant to an activity

5. Partners Area

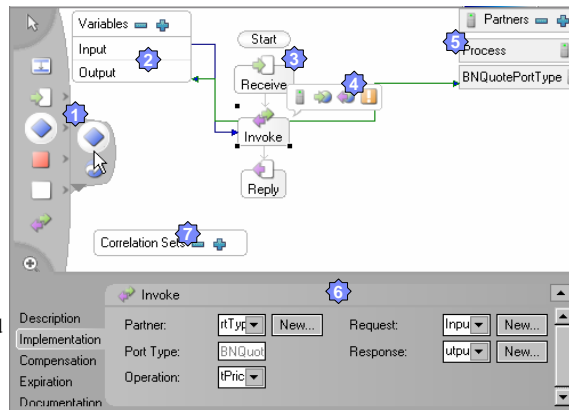
– Contains the process Partners and Process Interfaces

6. Details Area

– Displays properties that are relevant to the object that is selected on the canvas

7. Correlation Sets Area

– Contains the correlation sets are listed.



Continuing Service pain points

▪ Service Modelling

– Granularity

- coarse vs. fine grained.
 - What suites one business doesn't suit another

– Abstraction

- Differing bodies, Differing agendas. Standardisation of infrastructure not process/
 - Process model
 - Services
 - Operations
 - Types (OASIS UBL)

▪ Versioning

– Backward compatibility (Preserving investment, minimising change, improving ROI)

- Process
 - Version via the URL
- Structure
 - Version via Operation
- Content?

▪ State Management

– Different patterns means different policies.

- Synchronous
 - Client maintained non functional token, stateless except for non-functionals.
- Asynchronous
 - Client maintains state of functional token.
- Process
 - Client and Process maintain state. Process needs to timeout/compensate.



Resources / Questions?

- **Telstra Wholesale**
 - <http://www.telstrawholesale.com/linxonline/index.htm>
- **Perspectives on Web Services**
 - OOPSLA 2005 report: <http://www.perspectivesonwebservices.de/>
- **IBM DeveloperWorks SOA site**
 - <http://www-106.ibm.com/developerworks/views/webservices/articles.jsp>
- **There are many useful articles there. Highlighted here are:**
 - WSDL style/use: <http://www-106.ibm.com/developerworks/webservices/library/ws-whichwsdl>
 - JAX-RPC Handlers: <http://www-106.ibm.com/developerworks/webservices/library/ws-tipjax2.html>
 - SOAP Headers: <http://www-106.ibm.com/developerworks/webservices/library/ws-tipjax1/index.html>
 - Collections: <http://www-106.ibm.com/developerworks/webservices/library/ws-tip-coding.html>
 - Imports: <http://www-106.ibm.com/developerworks/webservices/library/ws-tip-imports.html>
 - Roundtrip: <http://www-106.ibm.com/developerworks/webservices/library/ws-tip-roundtrip1.html>
 - Versioning: <http://www-106.ibm.com/developerworks/webservices/library/ws-backward.html>
- **Contact**
 - Kerard Hogg kerahogg@au1.ibm.com
 - Peter Chilcott peter.chilcott@au1.ibm.com



Thank You!