The promise and pitfalls of implementing a Services Oriented Architecture

Presentation to Planning and Implementing Service-Oriented Architecture conference, Sydney, 15-17 March 2005

Chris Tham
Head of Architecture, Distribution
National Australia Bank

National

Agenda

- What makes SOA so attractive to an enterprise?
- The Myth of the "Silver Bullet" – Will SOA succumb?
- What will it take to REALLY implement SOA properly?
- Some potential problems from implementing SOA, and some ideas for avoiding them

National

Just what is “SOA” anyway?

a) “The best thing since sliced bread” – guaranteed to cure cancer, solve world hunger, and retire Third World debt
b) Something to do with SOAP and Web Services
c) What my vendor says will be supported in their “Next Generation” product
d) Yet another meaningless Three Letter Acronym, to be replaced by another fad in 3 years
e) It’s the same thing as (OOP, RPC, CORBA, CICS, …)
f) All of the above?

National

One definition (what my team came up with)

- A Service Oriented Architecture is an *application architecture* within which *key business functions* are implemented as re-useable *services* with well-defined, invocable, *interfaces*, which can be called in a defined sequence to form *business processes*.
- Note: the focus is on business processes, not object, functions, data, message … this differentiates SOA from previous paradigms such as object orientation

National

Why should we care? What makes SOA attractive to an Enterprise?

- **Easy for the business to understand**
  - The notion of structuring functionality as a set of "services" ties in very well with business process re-engineering, and product/services offerings, supply chain, etc.
- **Relatively easy to implement**
  - Doesn't require complete rewrite of all systems, tools exist to "expose" functionality from legacy applications as services
- **Broad (universal?) industry and cross-platform support**
  - Who would have thought the likes of SAP, Oracle, Siebel, IBM, Microsoft would one day sing the same tune?
- **Lowers (but does not eliminate) the barriers to heterogeneous interoperability**
  - It's (relatively) easy for a .NET client to invoke a J2EE web service, across multiple development and infrastructure environments
- **Doesn't require all existing applications to be thrown away or rewritten**
  - There are techniques to expose functionality from legacy systems as services

National

The challenges of implementing an SOA

- **"The difference between a vision and a fantasy lies in the execution"**
- **It doesn't matter how "neat" or "clever" or "righteous" the concept is if no one understands or is willing to change**
- **At the end of the day, it's about the people, not the technology**
- **Are you able to get the business to understand and support the concept?**
- **Can you get senior management buy in and commitment?**
- **Are you willing to spend the time to take everyone else along the journey?**
- **How do you address scepticism, the "not invented here" syndrome, and those who don't see any need to change?**
- **Is it about revolution, or evolution?**

National

The Myth of the "Silver Bullet"

Is SOA in danger of being perceived as a Silver Bullet?

- **A "Silver Bullet" is a concept/idea that promises to revolutionize IT (and cure cancer, … etc.)**
- **The IT industry is littered with the carcasses of "Silver Bullets" – none of these have become "pervasive" in the way their advocators originally prophesized, neither have them replaced older bullets:**
  - Computer language: COBOL, FORTRAN, APL, Ada, Pascal, C, C++
  - Relational data modelling
  - Data warehousing
  - Object orientation
  - Message queuing, object brokers
- **"Silver bullets" tend to fail because:**
  - Hype and unrealistic expectations about potential impact and benefits
  - Utopian ideals are seldom realisable
  - Many silver bullets are based on an "all or nothing" philosophy – the old way of doing things is invalid, and all systems need to be re-engineered
  - Lack of true "buy in" and "acceptance, lack of change management
- **Failures of execution are followed by a tendency to blame the concept ("I never thought it was a good idea in the first place") and as a result the Silver Bullet is tarnished**
- **Everyone then moves on to the next Silver Bullet**
- **As a result, there is deep scepticism at all levels whenever a new Silver Bullet is proposed. This can doom the concept even before it is implemented.**

National

Two Possible Scenarios at opposite ends of the spectrum

"Armageddon"

- There are hundreds, perhaps millions of web services in your organisation – nobody really knows.
- Mutiple versions of each service, but don't know which ones are used, and by what.
- Spaghetti Junction: No one understands the complexities of the relationship between systems.
- No one understands the services that support key business processes.
- If a business process breaks, nobody even knows which application to fix.
- Everyone is afraid to change anything because it might break something else.
- You are stuck on old versions of packaged applications because you are relying on outdated web services no longer supported by the vendor.

"Nirvana"

- The whole enterprise is implemented on a Services Oriented Architecture.
- All key enterprise business processes are modelled using BPEL.
- These are implemented using well defined services, with agreed and known usage contracts/service levels.
- Services are based on consistent XML definitions based on a common object model.
- Services are closely integrated with human workflow activities.
- Business processes and services can be "automagically" and dynamically deployed and provisioned on virtualized hardware.
- Full round trip engineering between service modelling and service implementation.
- All services are known and well defined.

National

How to avoid Armageddon and get closer to Nirvana

- **Evangelise**
  - Make sure the business and senior IT management understand SOA, and get buy in and commitment.
  - Even more importantly, make sure everyone else does!
- **Governance**
  - Think long and hard about balancing between over-policing and anarchy.
- **End to end linkage**
  - Work out how to link business process design to service definitions to implementation to deployment to operations.
- **Understand what your key vendors are doing**
  - What is their positioning on SOA? When and how are they going to implement it? Do they have any ideas/commitment on how to maintain/evolve service definitions across major releases?
- **Enterprise Architecture suddenly becomes very real, and important**
  - Who will maintain the models and linkages between them? Who maintains a catalogue/repository of services?
- **Track evolution of standards**
  - Don't get left behind, but at the same time don't be on the bleeding edge.
- **Build a competency centre**
  - Who maintains the standards? Who is responsible for providing guidance and mentoring to development teams? Who is responsible for the end to end view of key business processes?

National

Think of SOA as a "religion" (both in a good and bad sense)

- **It requires enough people believe in it and have faith in it to make a difference.**
- **You need:**
  - Messiahs – people who can evangelise the New Way and show the way to the Promised Land
  - Scholars/Monks – subject matter experts who can document good practices/knowledge and develop best practices
  - Priests – those who can lead others on the path to salvation …
  - Parishioners – a large body of people who believe and is willing to change their behaviour
  - A Church for the faithful
  - "Hell" and "purgatory" for sinners?
- **A "jihad" or "crusade" may not be the best way to convert others, although in certain instances it may make a difference …**

National