# White Paper: Service Oriented Architecture Strategies

By Rowan Mountford and Dr Ant Kutschera
July 2008

Over the past 5 years the phrase "Service Oriented Architecture" (SOA) has become a term commonly used throughout IT departments in the Enterprise. There are many definitions[1], but the underlying idea is nothing new – hiding implementation details (the "how") and providing interoperable interfaces which any part of the Business or indeed any Business Partners can plug into, to use the services "on demand". A service typically represents a particular business process within the company, or may indeed be an orchestration of a number of business processes put together. A related technology, namely Web Services, has played a big part in providing the implementation of such services since they too offer interoperability and abstraction away from technology specifics.

Companies traditionally providing a "service" to their customers (consulting, operations in terms of data centres, etc) have jumped on the band wagon and tried to cash in on the hype surrounding SOA by turning into "service providers", even providing "web services" albeit in a non-technical sense of the phrase.

So SOA has become important to modern IT and understanding it and the options surrounding it is important to anyone working with IT today. This paper discusses various strategies related to the implementation of SOA.

## *The Various Options*

Typically the starting point for an SOA will be a "green field" project, meaning that there is a budget for a brand new project and its architecture will need to be defined. Alternatively, budgets might be made available to upgrade existing systems (hopefully already 3 or more tier) in order to enhance their functionality, increase their maintainability and reduce future costs of keeping these systems running. Other reasons for going down the SOA route are that a business may want to restructure its IT in order to allow sharing of common functions throughout the business.

In the worst case, projects have taken an existing middle tier and wrapped it up in web services. A more ideal situation is for the IT department of an enterprise company to put an architecture board in place who will analyse the various overlapping functions of their business and its partners. This board will then define a set of core services used by two or more departments. The strategy of how to then implement a cost effective SOA is then their next topic, and is indeed the focus of discussion for this paper.

By the experience of the authors, two typical scenarios unfold. The first is for a central IT department to implement the services named by the board, probably based on a central framework[2]. Each business unit can help to specify the interfaces of the services that they will use, giving them a chance to define what they need out of the service. Now comes the question of whether the business is allowed to dictate this service in terms of how each business unit needs it to be implemented, or whether the central IT board dictates a generalisation of the service such that the business needs to adjust. Both these scenarios will be discussed below.

[1] Just search on http://www.google.com
[2] The subject of a further white paper, coming this way soon

The second scenario is where the business unit providing a "physical" service (say, accounting) to other business units becomes the owner of its own IT services and gets the budget to implement these services and offer them to other parts of the business or indeed business partners. Again, the interfaces of these services can be dictated by the business unit, or more centrally by an IT board with an overview of the service landscape.

The above is rather wishy-washy and high level. In order to bring it back down to the real world, where both authors prefer to work in their day to day lives, consider the following example. An insurance company, Cowboysure, protects its customers against fraudulent IT service providers, who cannot deliver IT projects on time, to budget, nor with the required quality. They have a sales department who have the following service offerings:

1) Get brochure
2) Get Offer
3) Purchase Offer

There is also the accounting department who have these service offerings:

1) Submit monthly accounts
2) Get Sales Report

These services are fairly broad and high level. But now comes the question of how Cowboysure should go about turning their service requirements into a reality. Based on the above discussion of strategies, we now have four possible ways to define and realise these services, namely:

A) Define the interface centrally (central IT architecture board) and implement it centrally too (central IT department),
B) Define the interface locally (by the business unit) but implement the service centrally (by the central IT department),
C) Define the interface centrally (central IT architecture board) but implement the service locally (by the business unit),
D) Define the interface locally (by the business unit) and implement it locally too (business unit).

In tabular form, these options would look like this:

| | | Interface Definition | |
|---|---|---|---|
| | | Central | Local |
| Implementation | Central | A | B |
| | Local | C | D |

This paper will now continue with an analysis of each strategy. It will finish with a recommendation, and discussion of a cost effective solution.

## *Centrally owned Interfaces versus Locally owned Interfaces*

Interfaces that are controlled centrally by an architecture board have the advantage that the board has a good overview of the SOA landscape and all its offerings. As the SOA offerings are increased and the landscape grows and changes, so the board can ensure services are kept up to date, increasing their usefulness to the business thus maintaining/improving their return on investment.

One problem that centrally owned interfaces have is that they may get generalised in order to suit all parts of the business. This may not suit the business unit who sits behind the interface as it may not model their business process accurately. The result will be that the business unit needs to reengineer its physical processes to match those modelled in the IT world. Typically customers who buy into large costly applications might be willing to do this as such applications may be based on best practises. However there will always be times when a business chooses not to fit in and to invest in making their business processes more specific to their needs. In such cases, a centrally managed interface would need to be split and the resulting services would contain the specialisations that are required. Remember however that IT is already expensive enough, and just because a service can be made more complicated does not mean it simply should be. Part of the analysis should involve considering the costs of a more complex service, not only in terms of development, but also maintenance.

Locally owned interfaces have the advantage that they will really fit the business unit well, however at the expense of the bigger picture which a central architecture board should have of the landscape. There is then the danger that similar services might be implemented by several business units, resulting in increased costs (implementation as well as maintenance – consider the danger that half of the business might use one service while the other half uses a second service).

Another danger of locally managed interfaces is that you can end up with each business unit calling other services in the landscape as and when they like. Without the central overview of the landscape and the associated planning, you can end up with a "point to point" architecture as illustrated below. To reduce communication paths, complexity and related costs, it is better to have a central body in charge of the interfaces, analogous to a service bus along which service calls can be conducted (see below). For these very reasons, Enterprise Service Bus (ESB) architectures have emerged in recent years.

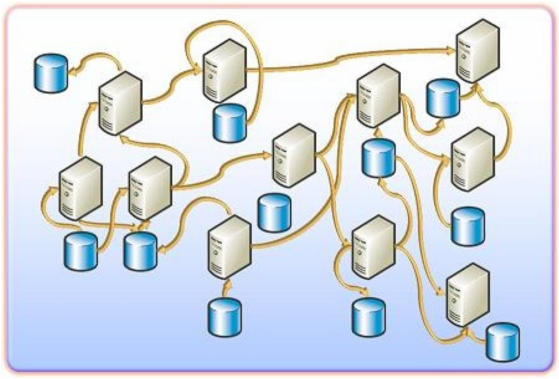

Figure 1: Diagram showing an example of a point to point architecture. Potentially, the number of lines of communications can be $n(n-1)/2$.

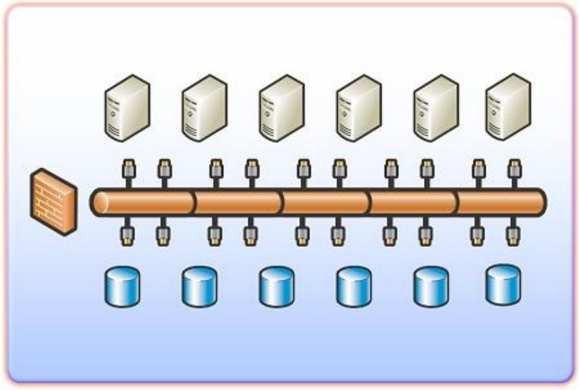

Figure 2: Diagram showing service bus, used to reduce line of communication down to *n*.

## *Central Implementation versus Local Implementation*

If services are implemented centrally, by a central IT department within an enterprise organisation, it is likely that they will all be based upon a single framework and the same implementation standards. But that does not necessarily mean that they will be more maintainable or implemented more efficiently. Those same frameworks and standards could just as well be passed on to business units who have their own IT departments, so that they can do the implementation themselves.

In fact the very nature of interface definition and implicitly, design by contract[3], is that once an interface is designed, the implementation that goes behind it becomes irrelevant, so long as the service delivers what is specified in the contract. So where the implementation is done – within the same department, within business units of the same company, or indeed by an outside company on the other side of the planet – no longer matters. It is only a question of whether or not the business who implements the services wants to structure their IT department so that it is a cost centre servicing all of the business centrally.

What however does happen, when implementations are managed locally, is that those implementations become decoupled. This might sound strange, but often when a single department develops everything, they start to share code and build in dependencies which do not belong. At the very most, centrally developed services belonging to different business units should never be more coupled than by the use of a handful of shared libraries. Those shared libraries need to be managed with their own release schedules and be completely independent of the services they support. Look to open source libraries[4] for good examples of how libraries should be developed and maintained.

The next problem that can easily occur when services are developed centrally is that services start being abused. Services which were originally designed to be an internal part of a decoupled service can be called by new services, even though that was never the intention. This happens equally with composite services (facades, higher level services) as well as atomic services (lower level, called by higher lever services). Once this starts to occur, the services are no longer decoupled and dependencies start making maintenance expensive. The author of such an internal service might make a change which then breaks the new service calling it. Since it was an internal service, there was never really any contract in place specifying exactly how it should work under all circumstances and as such the internal service has been abused. Modern service frameworks[5] have the notion of making such internal services "protected" such that they can control their visibility to other services. However decoupling the implementations, by allowing separate business units to develop them implicitly forces internal services to stay that way (internal).

## *Recommendations*

In terms of service interfaces, based on the discussion above, it is recommended that a central architecture board be put in place to handle the creation and maintenance of service interfaces. In cases where the business decides that a business unit should not generalise its services, the business may specify more complex service interfaces. However it should be extremely well documented which interface is to be used under which circumstances. Before the decision to specialise interfaces is made, an analysis

[3] http://en.wikipedia.org/wiki/Design_by_contract
[4] See http://commons.apache.org/ for examples
[5] For example, see Service Permissions under http://www.eclipse.org/equinox/

which includes development and maintenance costs of the specialisation should be conducted.

In terms of service implementations, based on the discussion above, it is recommended that services be independently developed for each business unit. Whether the implementation occurs within a central IT department or within local IT departments within the business, is irrelevant. It is far more important is to ensure that packages of services have public entry points and that their internal implementations remain hidden. These service packages should remain decoupled. The implementation should be based on the same frameworks and standards which the company should set in place centrally.

## *Moving Forward*

At some point, it will become time to get a Return on the Investment (ROI) made in defining interfaces, namely to implement those services, get the business and its partners to start using them and see success in terms of a more efficient business and a reduction in the Total Cost of Ownership (TCO).

The authors have seen such implementations fail on many occasions. The following is a list of steps to follow, in order to forego failure.

1) **Rapid Prototyping** Create prototypes which can be extended into fully working products. The prototype will prove that the service will work. If it is prototyped as a cut down version of a fully scoped service, it will then be fairly easy to extend that prototype into a fully working product, helping to reduce costs.
2) **Technology Stack** Use an existing and well understood technology stack which has many users and as such is tried and tested. Shy away from developing your own frameworks as they will simply add required effort to the project. Hardly ever does the case exist where the wheel has not already been invented, somewhere on the internet.
3) **Short Release Cycles** Release and test frequently. Not only does this reduce maintenance costs because bug fixing is cheaper the earlier it is done in the software lifecycle, but it also ensures the customer gets what they require, even if their original requirements were wrong and need to be re-specified.
4) **Business Ownership** Ensure the business is involved regularly and that they really want what is being supplied. If there is no business stakeholder willing to take responsibility for receiving the delivery, then it is probably not worth implementing the software.

This paper does not consider the technology required to move forward and that topic is out of scope for this paper. However, the authors intend to produce a further white paper with details on how to successfully move into this next stage of service implementation. Watch this space!

## *About the Authors*

Rowan Mountford works as an enterprise architect for Logica UK and has interest in SOA Strategies from his day to day work.

Dr Ant Kutschera has nearly 10 years experience of implementing software in the enterprise, in all aspects of the software life cycle, including but not limited to requirements gathering, architecture, design, programming, testing, delivery, support and maintenance. He currently works for various clients as an independent consultant, specialising in software architecture, Java EE, Rich Clients and SOA. His interests also lie in ESB / EAI. He can be contacted through whitepapers@maxant.co.uk.