Business-Driven SOA

Supply/Demand oriented SOA architecture - driving the SOA from a business perspective

Many organizations are now undertaking development of service oriented architectures, but the probability is that most will result in sub-optimal implementation. Most organizations will focus on a smaller set of objectives than they ought to, because they are overly influenced by technical concerns, and not sufficiently focused on the business service view. In this report we provide explicit process guidance on how to develop the right SOA for your business.

By Richard Veryard

Introduction

Problem Statement

Everyday practice is that projects take the easy way out – addressing a smaller set of objectives because it is too difficult to do the job properly. We observe many organizations now undertaking development of service oriented architectures, but the probability is that most will result in sub-optimal implementation because the relationship to the business requirements is not fully understood or communicated. Consequently business management will not see the value of the SOA.

Popular application packages claim to be SOA-friendly, but in many cases their web service interfaces simply expose the inherent inflexibility of the coding within.

There are many organizations that want to do SOA, but are stuck in the False Start position, as shown in Figure 1. Once you can break into the Upward Spiral (also Figure 1), the benefits of SOA reinforce themselves. But how to get there? This is a question of organizational change, and above all a question of IT governance – introducing a form of governance appropriate to SOA.

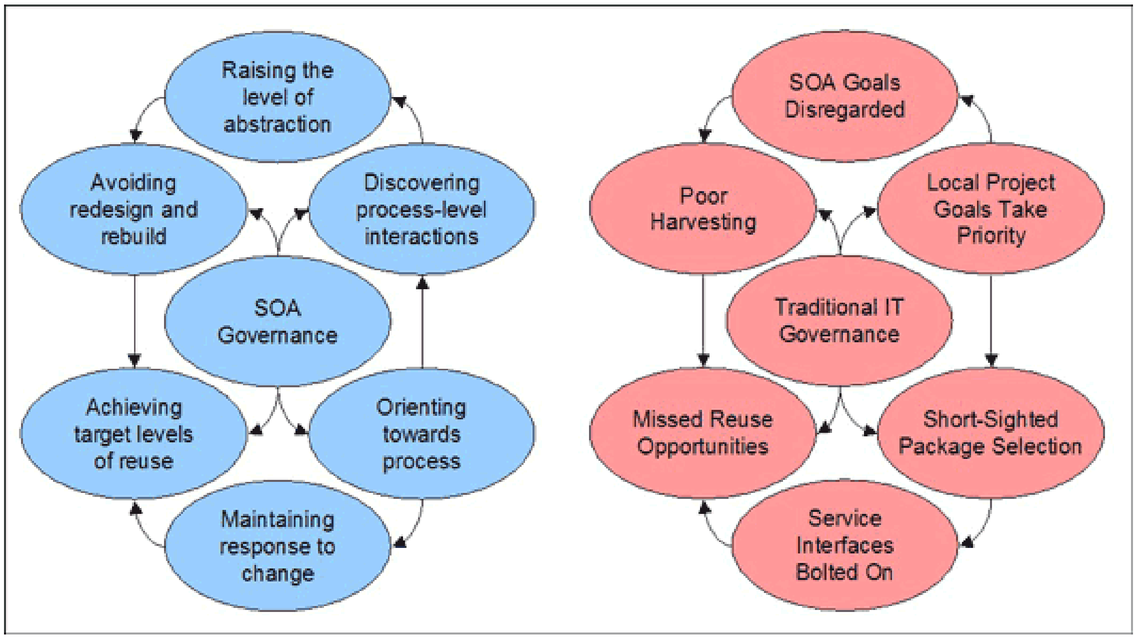
Raising the level of abstraction
Avoiding redesign and rebuild
Discovering process-level interactions
SOA Governance
Achieving target levels of reuse
Orienting towards process
Maintaining response to change
SOA Goals Disregarded
Poor Harvesting
Local Project Goals Take Priority
Traditional IT Governance
Missed Reuse Opportunities
Short-Sighted Package Selection
Service Interfaces Bolted On

Figure 1 - Upwards Spiral vs False Start

Example

Reuse	Use
Reuse is a software engineering metric – it refers to some notional saving in software development effort. If a component or service is called n times by other components and services, this gives a reuse factor of n (or possibly n-1). If a service is accessed in many different ways, this gives a higher reuse factor than if it is only accessed through a single channel. Multiple routes to a single service may increase adaptability, or it may increase complexity. Achieving reuse simply by multiplying the routes to a given service does not always deliver real business benefit.	Use is a business metric – it refers to the economics of scale and scope in utilizing a common resource. If a service is invoked n million times a month, it doesn't make any difference to the economics of scale whether the requests come through a single channel or many different channels. Counting use rather than reuse gives us a much better measure of the business value extracted from a given resource.

Box 1 Reuse or Use?

Here's a hypothetical example of what very commonly goes wrong with the reuse agenda. Let's suppose a project is building some new functionality, and needs to reference some employee context. Employee context is the responsibility of Human Resources (HR). In an SOA world, HR would ideally provide this context as a service. Furthermore, HR would have regard for its own economies of scale in providing this and similar services across the enterprise, and would establish this service at a level of generalization and abstraction that balances the immediate needs of the project requesting this service against the future needs of other parts of the company. This is what software engineers call **reuse**. See Box 1.

Fat chance. HR doesn't have the responsibility or the IT resource to do anything useful at all. Instead, the project that wants employee data has to go get it itself. This means either playing around with the HR applications (perhaps using screenscraping to avoid messing up the application code) or going straight into the HR database.

One result of this is that HR, despite being the formal owner of the employee data, is disconnected from many of the business processes that use and interpret the employee data. Data ownership becomes simply the administrative responsibility of populating a database and passively making it available, with little opportunity to monitor and control the appropriate use of the data. Furthermore, HR cannot then make any real changes to the employee data without affecting dozens of applications – and it may not even know what they all are.

If the HR function uses a well-known application package, we might think the situation would be better. Doesn't the latest version of the package promise full web-service enablement? And surely this means that whatever we want is already rendered as a web service?

Fat chance again. The package offers a limited range of interfaces. You may have to extract more data than you need (perhaps using multiple interfaces into multiple systems) and then process it in some complex way. You may have to try and make do with some data that are not quite what you want – and this may impair the effectiveness of the business process.

Meanwhile the application project is driven by the need to deliver a solution, and does not have spare resource to produce anything that is likely to be much use elsewhere. In some IT organizations, there may be a special group of developers who take certain pieces of development away from the solution delivery project and turn them into reusable assets. This may be done either preemptively (with reusable assets identified through the architecture function) or retrospectively (with reusable assets identified through harvesting).

From a use/reuse point of view, that's probably better than nothing – but it doesn't give us the necessary alignment between the production of data (HR) and the consumption of data (business process).

Meanwhile, there are some applications that reference employees – but it turns out that the full requirement is for a superset of employee – perhaps including temporary staff, staff of business partners, and other associates. These are maintained in separate data stores, not integrated into the main employee database. Thus an application providing office equipment may take a service request from any person legitimately working in the office – but should not need to know the exact status of the requestor, merely that the request is legitimate.

Ultimately, the right place for building and maintaining the HR service is HR itself. It is always the business that should be responsible for providing the service. If this service needs an IT implementation, the HR department may request an implementation service from the IT department, or from an external IT supplier.

When another department needs this HR service, the first contacts would always first be on business level, to agree quality of service, resources and other business issues. Then an IT connection service can be invoked (either by the providing department or by the consuming department) to make the necessary links at the IT level.

In the short term we may expect this linking to involve human intervention. While there are some increasingly sophisticated tools that support web service orchestration, these are at present only really suitable for use by IT personnel. However, there is clearly an expectation that this linking will evolve towards the automatic, especially for simple cases.

Business Organization Transformation through SOA

In this hypothetical example, we saw how the HR function was effectively disenfranchised from its own business data, and reduced (at least in this area) to playing a fairly bureaucratic administration role. This organizational pattern is typically found not just in HR but all over the large organization.

As Table 1 indicates, there is a parallel between the software problem of legacy and the organizational problem of bureaucracy. Business-driven SOA allows us to tackle either or both.

Software	Business	
Legacy Systems	Bureaucratic Organization	Inflexible, Unresponsive Possibly based on a historical attempt to make something automatic and efficient, but now perceived negatively
⬇	⬇	
Adaptable Systems	Agile Organization	Flexible. Smart. “On-Demand”

Table 1 SOA-Driven Change

To see how this transformation works, let’s continue with the HR example. If we take an architectural look at the HR function, we may see a variety of activities and other stuff – some of it meaningful and important, but perhaps some of it fragmented and bureaucratic.

One of the services that HR typically provides to the rest of the organization is recruitment. There may be different variants of this service – from the succession planning for a new CEO at one end to the annual recruitment of new trainees or the adhoc recruitment of temporary staff and subcontractors – but we may assume that there are at least some common elements.

This is a complex service that may run over an extended period (perhaps months) and may be decomposed into smaller services (e.g. multiple interviews with multiple candidates for multiple vacancies). While the line manager may act as the primary consumer of the recruitment service, the line manager may herself provide subordinate services (such as job specification and interviewing) as part of the orchestration of the recruitment as a whole service.

Let’s assume that HR is responsible for the whole recruitment service, and for the composition of this service from its composite parts, although the composite parts may be delegated to the line manager, or to external service providers (e.g. the recruitment agency).

But in some organizations, the HR function merely acts as an optional communication channel between the line manager and the recruitment agency, which line managers often bypass. The administrative work of HR is done by software packages, or by an external payroll bureau. There may be some expertise, but it is rarely if ever called upon. There may be some recruitment policies (including compliance with certain procedures), which HR tries vainly to enforce.

More generally, as we have seen, the HR function has abstract responsibility for managing employee relationships, including the holding of employee data to support a range of business processes, and overseeing a range of employment functions. In these areas, HR may be regarded as performing a series of governance services, responsible (among other things) for protecting the employee against unfair or arbitrary treatment, and protecting the company from employment tribunals and other legal action.

In many organizations, HR responsibility is increasingly pushed out to line management, leaving the HR department more than ever in an indirect supervisory capacity – orchestrating procedures and monitoring compliance.

Orchestration and compliance are of course both classic SOA stuff. But these alone possibly don’t constitute a sufficient justification for a separate Head Office department with its own seat on the Board – since they could reasonably be lumped together with a range of other internal audit and compliance functions.

The key SOA design principle here is that an organizational unit has to reflect a coherent chunk of service capability, and encapsulate some complexity (separation of concerns) on behalf of the rest of the enterprise.

So SOA business design tells us both whether HR has any meaningful coherence, and if so what its scope should be. This is based on a clustering of activity, capability and assets – or more generically, a clustering of practice and knowledge.

In the case of HR, the most likely clustering is one that is centered around a single key business resource – initially identified as “the employee”. By stepwise generalization, this may be extended to other categories of individual stakeholder, such as temporary staff, staff of business partners (who may be treated as employees for some purposes, including office space and access), and other associates. It may also include pensioners and their families. But where do we stop? Do we include shareholders, customers, and journalists? The answer to that question should come not from an abstract notion of generalization, but from a detailed and grounded look at the clustering of actual business processes associated with each category of person.

In other words, the clustering for this organizational function may be entity-based, as in this example, with a set of business processes that manage the lifecycle of a given entity. The appropriate scope of the entity is discovered through supporting a broad range of business processes, and by aligning the function as closely as possible to these business processes.

Not all organizational clustering will be entity-based, however. A sales management function will be centered around a coherent set of sales processes. Clustering here may address such issues as whether product customization and configuration should belong to the sales process or to the delivery process.

Where it gets even more interesting is where a responsibility is seemingly pulled into two different clusters. For example, one important overlap between the sales function and the HR function is in the matter of sales commission payable to individual sales executives. This has important implications for the achievement both of HR objectives (recruitment and retention of sales executives) and of sales management objectives (motivating and rewarding the attainment of sales targets) – and there is potentially a conflict between these two objectives. A higher-order process is required, which will moderate between the HR process and the sales management process.

There are lots of organizational design issues here that barely touch on IT itself, although they may be strongly influenced by the opportunities SOA provides at the IT level for simplifying administration, communication and supervision. In turn, when we are designing IT services to support the HR function, we may be influenced by the opportunities for radical organization redesign afforded by SOA, and ensuring that the IT services support the underlying business process, rather than the current organization structure.

A New Theory of Service-Based Design

The architect Christopher Alexander has had an enormous influence on software engineering for many decades. His first book “Notes on the Synthesis of Form” was referenced by the structured methods gurus of the 1970s, and his later work on design patterns has inspired a wealth of activity.

In 1987, Alexander and a few colleagues published an account of a design experiment under the title “A New Theory of Urban Design”. This book shows how order can be created without top-down planning, by a governance process that imposes some simple structural principles on all design activity.

There are some strong analogies between town planning and service-oriented architecture[1] , which make it reasonable to translate Alexander’s governance process into the SOA domain.

1. The distribution of design. Detailed design decisions are taken within different organizations, each following its own agenda (e.g. commercial or political goals).
2. The constancy of change. Elements of the whole are constantly being redesigned and reconfigured, and new elements are constantly being added. Structures must evolve in robust ways.
3. The need for progressive improvement. Each design increment should not only make local improvements, but should have a positive effect on the whole.
4. The recursive nature of the architecture. Similar design tasks must be carried out at different scales (levels of granularity).

Alexander formulates the following highly generalized principle: “Each new act of construction must create a continuous structure of wholes around itself.” Alexander then spells out what is meant by the word “whole” in an architectural context. One of the key consequences of his notion of wholeness is that a designer tasked with constructing something of scale X must pay attention to three scales – larger than X, same as X, and smaller than X. Part of the governance process is to ensure that there is a reasonable balance of projects of different scales.

Following Alexander, SOA designers need to be aware of the following three design rules.

1. Each service should contribute to some larger-scale “whole” service.
2. Each service should have clean boundaries with other services of the same scale as itself.
3. Each service should be supported by a series of smaller-scale services.

Wholeness can be understood in terms of rotational symmetry, as shown in Table 2.

Service Type	Basis of Symmetry
Entity-Based	Life-Cycle – including all events that alter the identity or status of the entity
Process-Based	Closed Feedback Loop – including all activities that establish stability / equilibrium between independent elements.
Rule-Based	Open Learning Loop – including all activities that allow the rule/policy to be reviewed and improved

Table 2 – Rotational Symmetry

Let’s now look at the HR example in the light of this service-centric process.

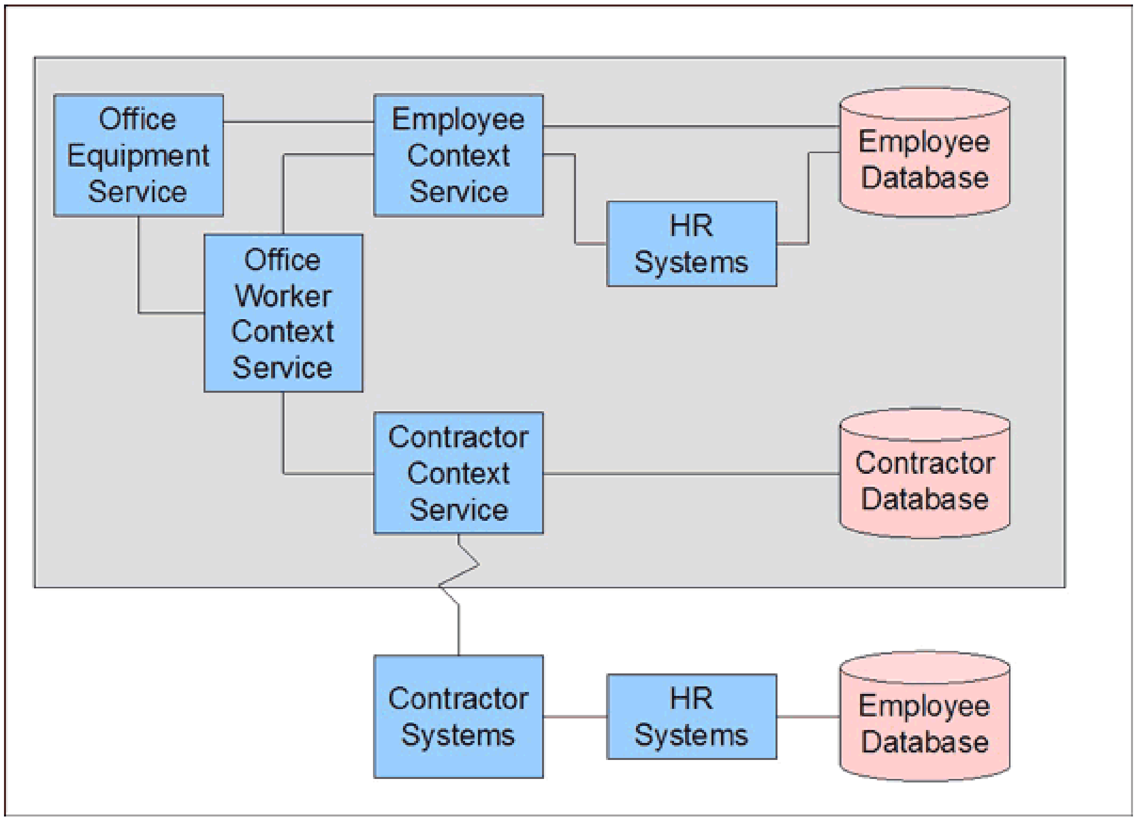

Figure 2 Incremental Improvement of HR-related Services

Step	Status	Consequencese
1	The office equipment application obtains employee data direct from the employee database.	The office equipment application is given direct access to employee data. This means it has to be subject to higher levels of privacy and security protection than it otherwise needs. And it is affected by any local changes in the HR database systems.
	The mechanism for providing office equipment services to non-employees is not properly defined.	Provision of services to non-employees is adhoc and often inconsistent. Sometimes non-employees get a better service than employees, sometimes they don’t get any service at all. In a few cases, non-employees have been added to the HR employee database, with status codes to denote that they are not real employees. This makes the HR systems more complex. There is a risk of processing error if HR applications fail to check the status code properly.
2	An employee context service is introduced, to shield the office equipment service from the HR database. This initially gets data direct from the HR database. When appropriate web services are provided by HR systems,	This service can then be used by other applications/services as well, thus gradually reducing the number of applications/services with direct access to the HR database.

	these will be used instead.	
3	An equivalent service is created for establishing Contractor Context. This initially gets its data from a local file of approved contractors.	We can now clean the non-employees out of the HR database, which makes it simpler and more reliable. The contractor context service allows a range of office services to be provided to contractors in a consistent and controlled way.
4	The Contractor Context service now gets its data by remote access to the contractor's systems.	This means that if a member of contracting staff changes employment status with the contracting company, this can be immediately reflected in office services.

Table 3– Incremental Improvement

In this incremental improvement, we have strengthened (by simplifying) the lifecycle of EMPLOYEE, created a lifecycle for CONTRACTOR (with specific reference to events that affect the status of contractors), and started to build a larger supertype OFFICE WORKER. The lifecycle of contractor can now be understood as an intersection between the local lifecycle of CONTRACT and the remote lifecycle of EMPLOYMENT within the contractor's own systems.

Each of these services may use smaller services, and contributes to larger services. The office equipment service is linked in turn to a series of business processes that require office services.

The ultimate touchstone of service wholeness is the business process. Our SOA design process is both service-centric and business-process-oriented. It allows SOA structures to be evolved incrementally, through SOA design and SOA governance.

Scope of SOA Governance

In a true SOA world, the supply/demand relationship is a dynamic one, and we should be wary of institutionalizing this into a fixed twin-track organization. In order to maintain true adaptability, we have to have a form of SOA governance that is itself adaptable.

Governance is not just about planning. (Governance that focuses on planning and neglects the rest of the life-cycle is sometimes referred to as "govern-once".)

And governance is not just about the inhouse IT function – although that's an obvious place for the CIO to start – get your own house in order, before tackling broader governance issues. A broader framework for governance is shown in Figure 3.

Software Governance Coordinating software development, acquisition and (re)use across internal and external domains to achieve maximum agility and economics of scale/scope. Monitoring the technical performance of software services, including security.
Business Governance Coordinating business development, negotiation and collaboration across internal and external organizations, including trust. Monitoring the business performance of services, including ROI.
Service Governance

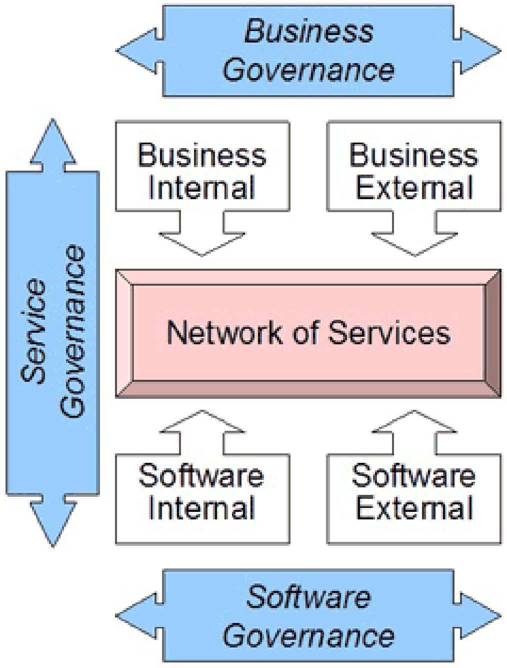

Figure 3

Aligning software governance with business governance.

Roadmap

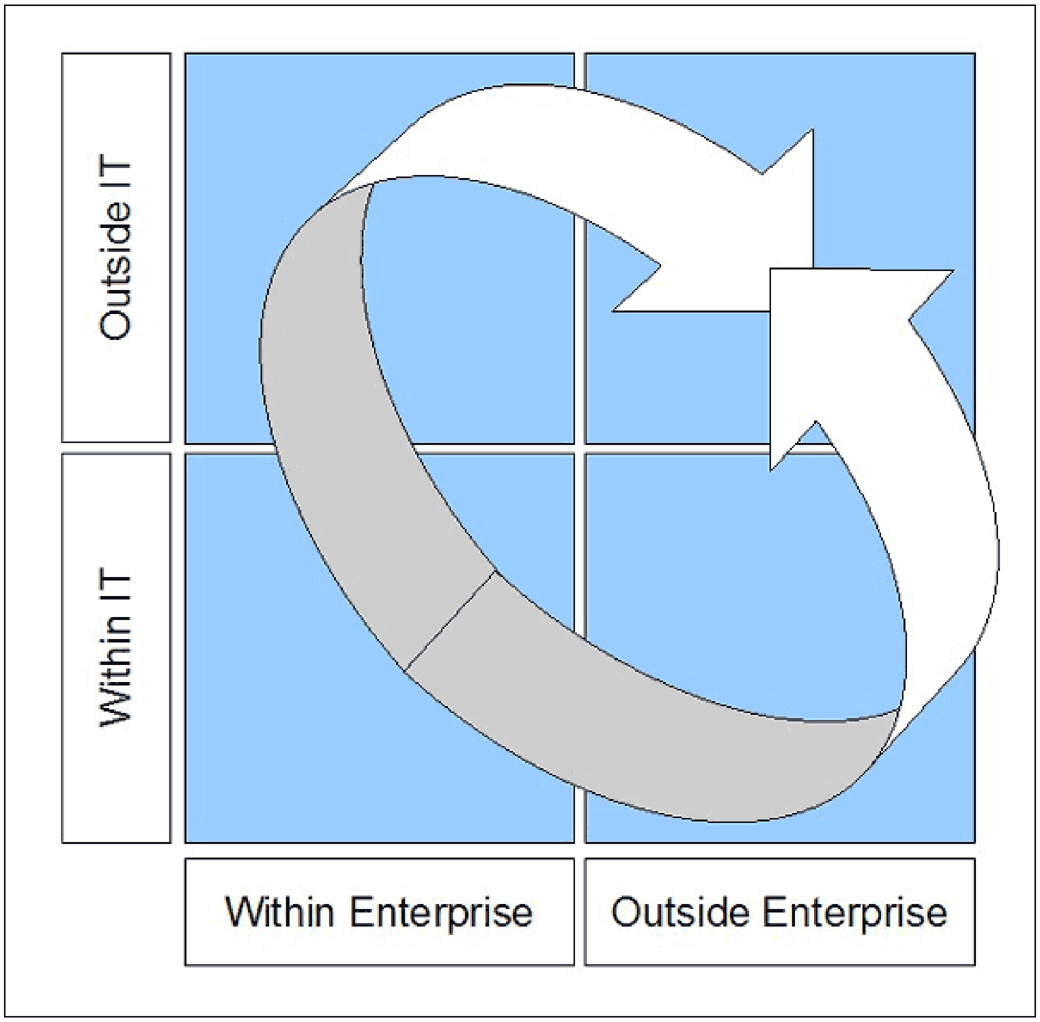

Figure 4 Roadmap for SOA Governance

The roadmap for SOA governance is in several steps. In step one, SOA Governance is primarily a matter for the IT organization inside a single enterprise. In the final (more radical) step, SOA Governance extends beyond IT, and beyond the boundaries of a single enterprise. There are two alternative paths, as shown in Figure 4 you can go up and then across, or across and then up.

Thus in the HR example, we saw how HR services may be managed. In step one, HR services are managed by some HR team within IT, representing the HR organization. At a later stage, HR services are managed directly by the HR organization, calling upon services from IT as required.

At this point, the design of the whole enterprise, as well as its relationships with the outside world can now be organized along SOA principles.

However, most organizations will not be willing to take this more radical step until the earlier steps have been mastered - perhaps several years hence. However, there will be some organizations that may be forced to broaden the scope of SOA Governance much sooner than this.

Acknowledgements

Thanks to Rob Mocking of KLM for initiating this discussion, and for some great ideas.

CBDI Material

1. For a comparison between urban planning and IT, see Pat Helland, Metropolis, in Microsoft Architects Journal, April 2004.