

Mind the Gap

What is SOA and what's missing?

SOA (Service-Oriented Architecture) has become a near mantra for IT strategy advice in recent times. In this paper we take a step back, looking at the pressures being placed on IT and the required characteristics of its applications. We then examine what SOA is and how it can help, but also highlight what we term "the SOA Gap" – what's missing from current technology offerings. Finally we outline an incremental approach to filling that gap in transitioning to SOA.

June 2005

by John Cheesman and Georgios Ntinolazos, Strata Software Ltd

Business and IT Agility

Today's enterprises are highly automated but exist in an ever more competitive environment. To compete they need to be able to provide new and improved products and services to their customers quickly and efficiently – in short, they need to be *agile*.

The IT department in turn has to respond to this requirement by improving the ease and speed of application development, integration and maintenance – IT itself must be agile, and so must the applications it delivers.

Agile Applications

We may characterise an enterprise's applications as a layer between business and technology – applications exist to automate business processes by applying technology.

The application layer has a rather difficult job. Rather like a tectonic plate boundary, it has to bridge two worlds that are constantly changing or under pressure to change and which have a degree of inertia and a lot of momentum. It therefore needs to exhibit some important characteristics:

- **Business Flexibility** – new applications can be created, updated or re-configured to support new business processes as required.
- **Technology Flexibility** – the dependency relationships between the application-layer services and the implementing systems and technologies needs to be manageable and flexible. This characteristic is sometimes termed "platform-independence".

SOA – the next step

Service-oriented architecture (SOA) defines a set of characteristics and features for the application layer. Many of these are not new, but have only been partially achievable with existing technologies and standards. However, there is now increasing standardisation and acceptance around a number of technology innovations which has significantly increased the momentum behind a service-oriented approach.

The basic principle of SOA is that the application layer be organised as a collection of "services" which are then assembled or orchestrated as required to provide the same function that applications and systems provide today – business process automation, but in so doing, enable better flexibility and integration. Such an approach means that the fundamental unit of both IT delivery and internal communication, its *lingua franca*, changes from the application to the service.

We define SOA as follows:

SOA is a platform-independent architectural framework for business process automation through the assembly of business services.

Architectural flexibility comes from (i) the approach of assembly from parts and (ii) the "pluggability" of the parts themselves. So what are these parts and what makes them pluggable?

A business service is an independently deployable set of business operations and their associated data that is: distributed, self-describing, encapsulated, flexibly-coupled and designed to be reusable.

We use the term "business service", rather than simply "service", to emphasise the operations that have business meaning and to distinguish them from technical or infrastructure services.

Figure 1 shows the broad shift in architecture viewpoint from an organisation and integration of systems to the assembly and integration of services.

Isn't this CBD?

Many of these characteristics have been promoted before under the heading of Component-based Development (CBD). What's different now?

SOA builds on CBD – the two disciplines have many aspects in common such as distribution, encapsulation and reuse, and both place importance on separating specification from implementation.

But there are a number of significant new dimensions too:

- **Flexible-coupling:** Business services are designed to support a variety of interaction mechanisms and protocols from loosely-coupled to platform-specific as required.
- **Runtime emphasis:** SOA focuses on integration and *service* reuse at runtime, whereas CBD emphasises *software* reuse at design time. These are complementary but distinct viewpoints. In particular, a service is a runtime object, typically with state. Service reuse and integration therefore concerns the integration and reuse of business data and its associated business rules, not just the reuse of software.

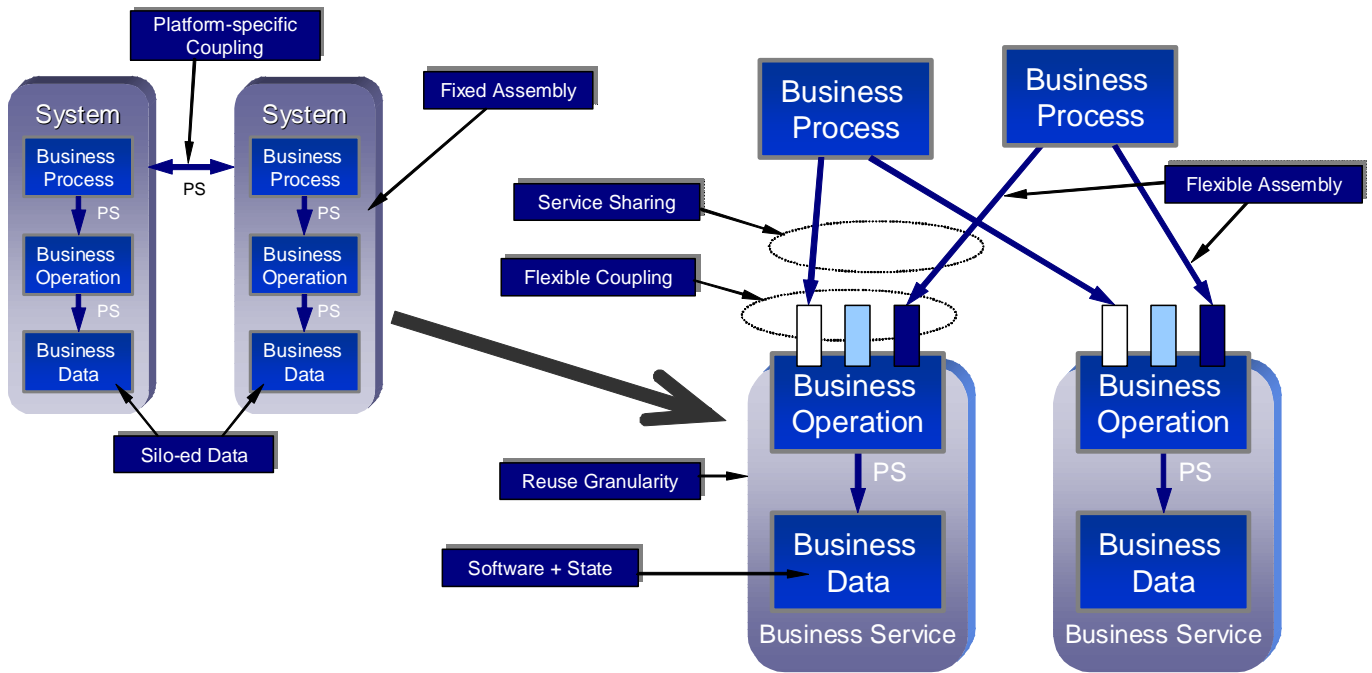


Figure 1 - System integration becomes Service Integration

- **Service discovery:** SOA assemblies can be formed either statically at design time or dynamically at runtime. This increases the importance of providing self-description data (known as meta-data) to support reuse decisions.
- **Role distinctions:** the traditional Architecture, Provisioning and Assembly roles become even more distinct than with CBD. The Architect defines service granularity and coupling policies, the Provisioner needs to surface existing systems and software in those forms, and the Assembler manages the runtime realisation by selecting the required services to assemble, and applying the appropriate architectural policies to that selection.

The SOA Gap

As with virtually every significant development in the industry, the advent of (justified) interest in a particular area or approach pushes technology opportunities to the fore and the necessary application-layer best practice becomes established in its wake. SOA is no exception. One part of the “SOA Gap” is technical, the other is what we might term “organisational”.

On the technical side the SOA gap is the current absence of, or lack of standards for, concepts, patterns and methods which apply at the *application*

level as opposed to the technology level.

For example, Web Services are a technology development which enables loosely-coupled, platform-independent service interaction. But when should they be used? How granular should a service be? How many different types of service are there?

Similarly, take Business Process Execution Language (BPEL) which allows the flexible orchestration of services. This holds out the prospect of a simplified and more flexible process automation approach. But what type of business process should be automated this way and at what level? What granularity of service should it access? Are business processes themselves services?

These concerns can be summarised as follows:

- (i) the lack of a standardised concept set and vocabulary for applications and services and their relationships and lifecycles – an Application Reference Model for SOA;
- (ii) the lack of SOA methods and techniques which apply these concepts, and consequently;
- (iii) the lack of tool support in the development process for the business service concept and its lifecycle.

On the organisational side, the SOA gap is somewhat similar to certain adoption barriers encountered with

CBD, and if anything, can be more significant than the technical challenges:

- The ROI case for moving to SOA can be difficult to make within organisations which are project-funded. This is because SOA benefits accrue across projects rather than within them. An ROI case therefore needs to be made at a cross-project level, normally for a business domain, and budgets allocated accordingly. Existing business case processes and organisational structures may impede this.
- Process changes – the basic life-cycle unit of the development organisation becomes the business service. This means that application development and integration processes, configuration management processes and, to some extent, project management processes, all need to align with this.
- Responsibility changes – service development and maintenance ownership boundaries will typically be different to application ownership boundaries. These differences may require responsibility changes, and possibly organisational changes, which, again, may be impeded by organisational inertia.

Fixing the Gap

How do we tackle these technical and organisational issues?

Figure 2 depicts a basic approach to getting started. It separates two distinct "routes" to SOA. The exact order of play will obviously depend on the organisation, but this is an example approach.

Although SOA is sometimes positioned as a revolution, in terms of business and IT change it is best adopted in an evolutionary and incremental manner – start small and local, get some experience and comparative metrics, gain confidence, then repeat.

This approach can best be initiated through opportunistic service harvesting on an existing or planned application development or systems integration project, or possibly on an investigative pilot expressly for that purpose. Such projects allow the creation of one or more initial services within a well-defined area and under the control of the application architect.

Once experience has been gained, and there are real services in the catalogue, everyone in the organisation can make reference to a common example and some concrete metrics and measures – things move from theory to reality.

Having some actual services and some measurable experience of developing and using them then facilitates the development of a business case for a more strategic move to SOA, and the creation of an explicit portfolio service planning activity.

As strategic planning develops then the appropriate technical, process and organisational changes can be piloted, established and tuned.

On the technical side it is critical to develop an application reference model for the organisation, including a **business service standard** and its associated life-cycle processes (architecture, specification, design, provisioning, publication, assembly, deployment, discovery, versioning and so on).

This establishes a blueprint for the application layer to develop and enhance over time and forms a key part of an enterprise's SOA reference model.

Harvesting a business service from one or more existing systems is a good way of establishing an initial business service standard since it will quickly yield something that reflects the need of current business processes. It will therefore contain all the dimensions needed and be richer than creating a new service from scratch. Once the main artefacts and standards exist, then the processes and automation around them can start to be formalised.

Harvesting a service from a system involves understanding the different roles of the system in its various interactions in support of a particular process or use case. The information requirements of each role and each interaction can then be assessed and normalised to provide first cut service specifications.

Summary

SOA defines a new approach to the organisation of application software which holds out the promise of improved IT agility and increased business agility.

However, in the industry at large the SOA message is accompanied by the usual clamour of technology and product specifics from vendors. Many of these offerings do have a contribution to make, but throwing technology alone at a problem doesn't generally yield the required results.

By recognising the SOA gap and fixing it through the development of the appropriate organisational and technical standards and structures, the power of the emerging SOA-enabling technology can be put to good use, and the promised benefits realised.

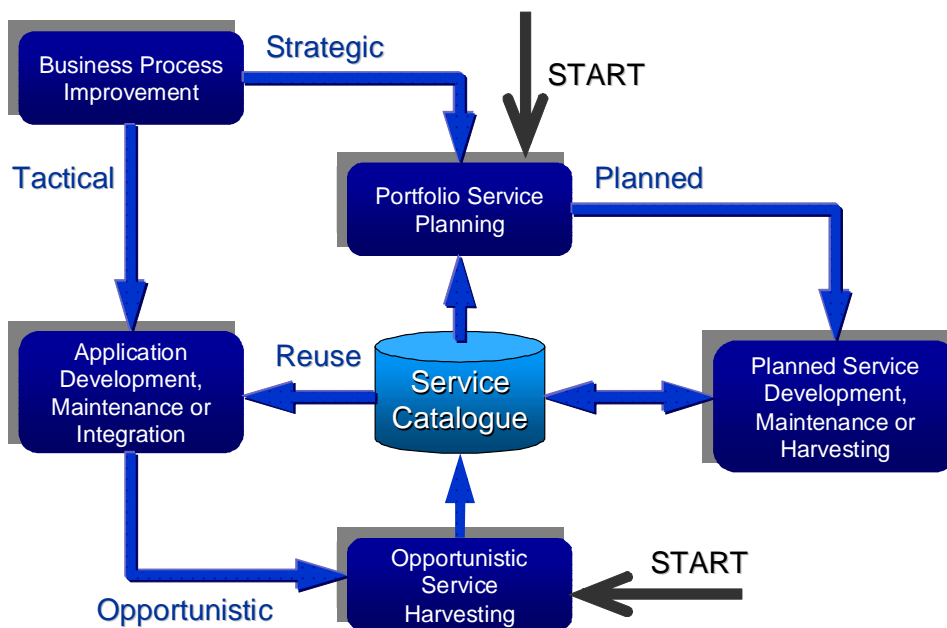


Figure 2 – Routes to SOA

Strata Software

Strata Software provides consultancy, training and products to support application development productivity and service-orientation. The Strata SIGMA™ approach unifies the business and IT domains around the Business Service construct, and applies an architectural framework to the business service lifecycle.

Providing Business Agility with IT Productivity