mwd

macehiter ward-dutton

Application delivery and SOA: a lifecycle approach

Neil Ward-Dutton and Neil Macehiter, Partners

November 2005

Software services have the potential to provide a common unit of design in both the development of software, and its management and monitoring. What's more, a service-oriented to application delivery approach "done right" can significantly improve software reuse, flexibility and comprehensibility, and also improve the visibility of software's value to the business. As a result, SOA can help address the needs of both business and IT stakeholders: acting as a unifying force in IT-business alignment.

However, realising that potential requires an end-to-end approach to software development and management based around a clearly-structured application delivery lifecycle which recognises the roles and responsibilities of the key stakeholders; managed by a well-defined governance framework; and supported by tools and technologies which promote collaboration across the lifecycle.

This paper explains the key issues which organisations pursuing SOA initiatives need to tackle in order to establish fit-for-purpose application delivery capabilities.

Macehiter Ward-Dutton is a specialist IT advisory firm which focuses exclusively on issues concerning **IT-business alignment**. We use our significant industry experience, acknowledged expertise, and a flexible approach to advise businesses on IT architecture, integration, management, organisation and culture.

www.mwdadvisors.com

Why SOA?

Today's business environment demands an evolved IT approach

Globalisation of markets is driving customer choice to a point where competition for custom can be extreme; driving overall business success to depend not only on internal performance, but on performance of actions coordinated throughout complex value chains. Stiff competition also means that business success is often highly dependent on consistent customer service excellence – and therefore a customer-centric perspective which depends on the overall performance of the service delivered to the customer, which spans organisational functions.

Meanwhile, the growing adoption of the outsourcing of IT, and increasingly of entire business processes, together with accelerating use of offshore service providers, demands the integration of these outsourced services with the rest of the organisation. At the same time, long-term successful relationships with outsourcers demand that it must be possible to measure and enforce the quality-of-service and commercial aspects of their service provision.

Furthermore, the ongoing harmonisation of regulation, which is occurring against a backdrop of local regulatory demands, increases the burden of compliance, which is driving IT and business practices to become further intertwined. To efficiently alleviate the burden for the long term, it is best to assess compliance implications in the context of business processes, and only then enhance the IT capabilities which underpin those processes. Efficiently implementing automated support for regulatory compliance demands flexibility to increase responsiveness and localise the impact of change, together with automation of compliance status monitoring and reporting.

SOA can enable the needed business-driven balance between IT flexibility, efficiency

Whilst the details vary, all these broad business challenges point to an increased focus on service orientation within the business, and on service-oriented IT which is optimised to serve the needs of key business processes end-to-end. However where commercial pressures demand a business-driven balance between flexibility and efficiency from IT, the reality for most organisations is that their IT portfolios are currently incapable of responding to the challenges which this brings. Information to support decision making is abundant, but it is often difficult to access; new IT systems and applications are difficult to integrate; resource utilisation is sub-optimal, with significant redundant capacity; and IT solutions are too inflexible to support business change. In short, it seems that very few enterprises' IT portfolios, strategies or delivery capabilities are well-aligned with the requirements of their business units.

Improving "IT-business alignment" requires an evolution both in technology and technology thinking, which makes it easier for organisations to more effectively exploit their existing IT assets; and re-prioritise IT investment, delivery and strategy so that IT assets can be more readily directed to deal with business change. This is where SOA comes in, because the overall approach described here is most effectively supported by a service-

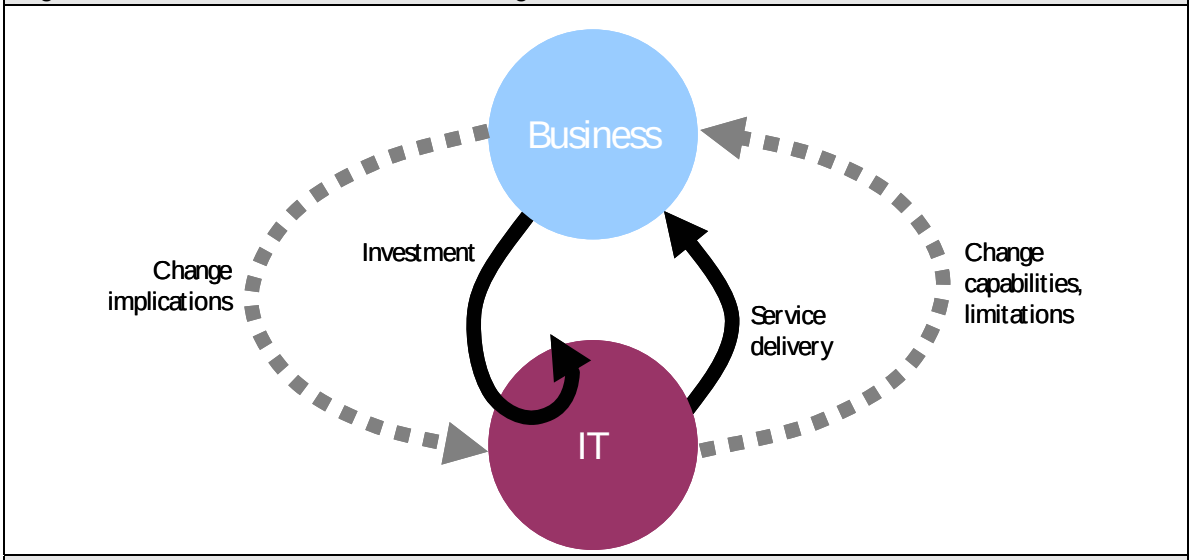
oriented model for enterprise software.

How can SOA improve IT-business alignment?

A simple IT-business alignment model

In our work, we have come to see IT-business alignment as an ongoing *process*, through which IT delivery organisations and businesspeople work together towards a situation where IT capabilities and limitations are influenced by, and influence in return, business priorities and strategies – see figure 1.

Figure 1: The elements of IT-business alignment



There are three important elements in IT-business alignment: investment, service delivery, and collaboration in change management.

Our model of IT-business alignment is based around three aspects of the ideal relationship between an IT delivery organisation, and its "customer". The three relationship aspects are:

- **Investment**. IT is now integral to the way that businesses operate – but historically, IT has been viewed by the business as somewhat of a "black art", best left to IT practitioners, with the result that investment in IT has been treated as a special case. With IT now playing such a critical role, this position is no longer tenable. Investment in IT must be subject to the same priorities which govern investment in other assets – people, facilities, production lines etc. – on which the continued operation of the business depends
- **Service delivery**. The way in which the IT delivery organisation provides services to the business must be governed by the same business priorities as is IT investment. This extends to encompass the way that IT service delivery is measured. Only then will it be possible for businesspeople to assess the business return on their investment in IT
- **Change**. With IT now so integral to the way that business operates, it is no longer feasible for business decisions to be taken without a clear understanding of the IT

implications of those decisions. Business leaders and the IT delivery organisation must participate as peers in the business change management process and adopt a systematic approach to assessing the IT implications of any change. Such collaboration will put the IT delivery organisation in a position where it can actually influence business change by highlighting the challenges and opportunities arising from technology change.

SOA can support all aspects within the model…

A service-oriented approach to delivery of IT solutions has the potential to significantly boost all three key aspects of the IT-business relationship described above. There are four major technology-related benefits of SOA that have the potential to play here:

- **Large-scale reuse**. This is one of the most-talked about benefits of SOA. With effective service-based software reuse programs in place, IT delivery organisations can build up libraries of business-meaningful functionality that are not tied to particular usage scenarios, and (crucially, unlike previous attempts at object-based reuse) are easily composable and re-composable to meet new business requirements. With these libraries, organisations can reduce the investment required to address new business software requirements; make delivery of new solutions quicker and more reliable; and also improve the accuracy and speed with which solutions can be changed
- **Improved flexibility**. Along with reuse, flexibility is the other commonly talked-about SOA benefit. Flexibility of course concerns the ability of solutions to be altered in the face of changing business and technology requirements, and is boosted by the loosely-coupled nature of the services which are composed to meet solution requirements in a SOA environment. Flexibility is a key to the change management element of IT-business alignment – but other aspects of SOA are important here too
- **Improved comprehensibility**. Successful SOA implementations create groups of services which can be clearly linked to individual tasks within business processes, as well as lower-level technical services. In other words, strong service portfolios have many key elements which are easily comprehensible to business audiences (examples might include services which manage customer information, or which validate orders). Software comprehensibility is not often talked about as a benefit of SOA, but it has a massive impact on the ability of a SOA initiative to improve IT-business alignment. When software functionality is easily comprehensible, it is much easier to build a common language between business and IT – which makes it easier to engage business stakeholders in real investment discussions and solution design work; and to show how particular "pieces" of software contribute to business activity
- **Improved visibility**. Beyond providing more comprehensible software solutions, successful SOA implementations also have the potential to increase the visibility of IT's value to the business. At the heart of this possibility is the fact that the idea of a "service" is both a business-meaningful unit of software design when a solution is being conceived and built; and a business-meaningful unit of reporting when a solution design is in production. For the first time, with SOA, we have the ability to clearly relate the production of solutions in an IT environment, to the operation of solutions in a business environment. Together with the promise of comprehensibility, the visibility that comes with SOA enables business stakeholders, administrators, designers and developers to share a common view of solutions which *makes sense to*

all of them. This has clear implications for more "business-aligned" IT investment and IT service delivery, and also in the cooperative management of change.

...but only if you implement a lifecycle approach

Maximising the reuse, flexibility, comprehensibility and visibility of software in business solutions are all possible within SOA initiatives – but making any of them possible will be very difficult indeed for an organisation that does not take steps to understand, formalise and manage the end-to-end lifecycle of services and the applications into which they are composed. Without a "joined up" approach to the analysis of requirements, and the design, deployment and management of application solutions, services made available for reuse will not be effectively located, understood or used; and the need to flex those solutions in line with changing requirements will not be clearly transmitted to the right people at the right time. Just as importantly, though, the visibility and comprehensibility of value of solutions both rely on effective collaboration between a range of people, roles and technologies throughout the lifecycle of solutions.

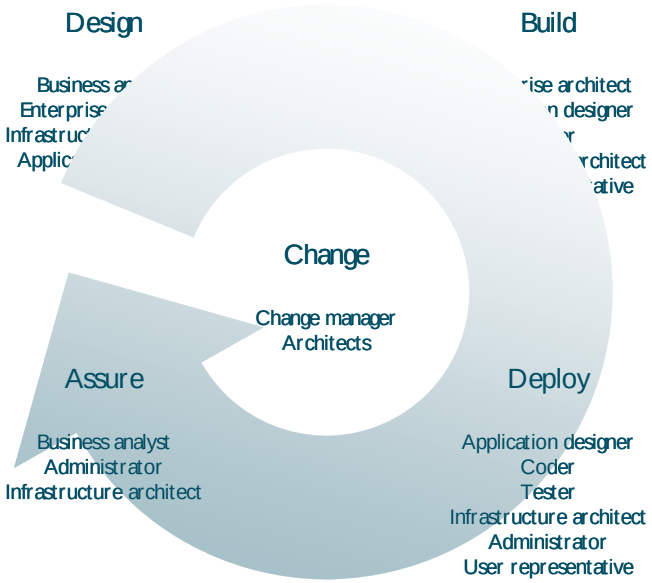
An application delivery lifecycle model for SOA

Here, we first discuss the high-level structure of the application delivery lifecycle model, the roles within it, and the capabilities which must support it. Following that are five sections, each of which examines a phase of the lifecycle in more detail – with an emphasis on the main steps involved, the roles which contribute, and the capabilities which support it.

Five main phases, with many contributing roles and capabilities

Our model of an application delivery lifecycle for SOA contains four main "doing" phases: design, build, deploy, and assure (as shown in figure 2). However SOA is an architectural approach which works well where change is a fact of life. This means that in addition to these main phases, there is a fifth, and central, element of the lifecycle which cannot be ignored: change management.

Figure 2: Phases in the SOA application delivery lifecycle



An application delivery lifecycle model for SOA involves resources playing a large number of roles. Management of change must be at the core of the lifecycle.

The architect role is the linchpin in the lifecycle

As figure 2 also shows, each phase in the lifecycle requires cooperative contributions from people in a variety of roles, from the business as well as across IT delivery functions. The key roles throughout the lifecycle, though, are the architect roles.

An IT architect's job is to work with a range of stakeholders to bring IT and business needs

together in the development and implementation of IT solutions. In this, the architect's natural value is to balance the need for good short-term business outcomes with the need for long-term high-quality IT value. In the context of a delivery lifecycle influenced by SOA principles, given the goals of maximising reuse, flexibility, visibility and comprehensibility and through these, improving IT-business alignment – this means that the role of the architect is absolutely crucial. It is the architect roles which are best placed to take responsibility for the effectiveness of the overall delivery lifecycle: helping ensure that information flows from one phase to another, and also ensuring effective communication between business stakeholders and the various IT roles.

We identify two types of architect in the SOA application delivery lifecycle: the *enterprise architect* role, which is responsible for maintaining a broad view that maps the organisation's business processes and priorities and relates these to the organisation's overall IT portfolio; and the *infrastructure architect*, which is primarily focused on formalising a view of, and rationalising, the organisation's overall investments in software, hardware and network infrastructure.

Supporting capabilities have to be integrated and intuitive to access

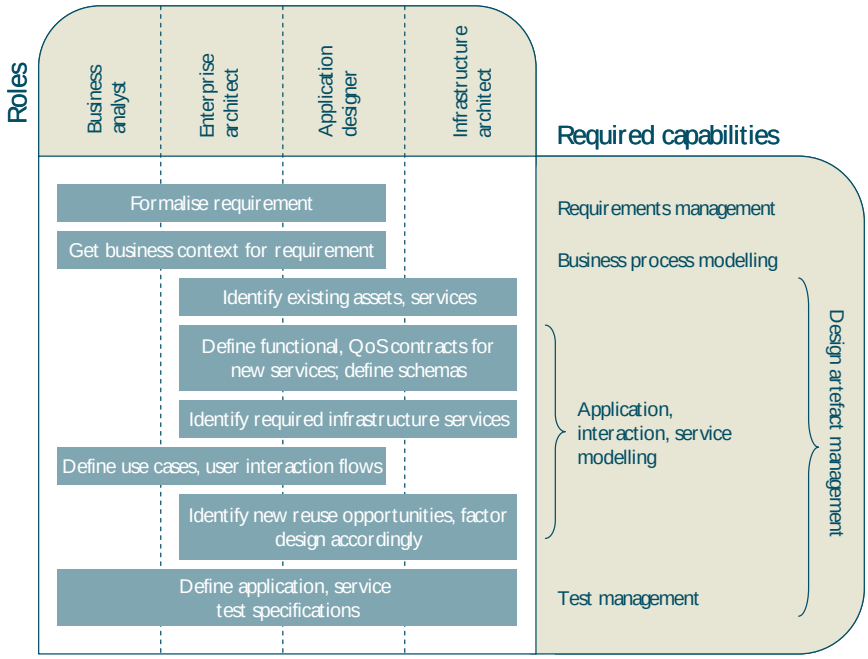
Software reuse, flexibility, comprehensibility and visibility can be improved through a SOA initiative, but none of these come from technology alone: the unfortunate truth is that all these require cultural adjustment, and focused effort to achieve. With a poorly supported delivery lifecycle, it will seem quicker and easier to create software solutions with no consideration of reuse, flexibility, comprehensibility and visibility, than it will to create solutions which reflect those goals. The multitude of technology capabilities that are needed to support the different phases of the SOA application delivery lifecycle – from requirements management and design environments, through to monitoring tools and optimisation feedback mechanisms – will only increase the natural inertia to process and architecture change, unless the tools which provide those capabilities:

- are integrated, in that they can pass key process artefacts between them intact
- are intuitive and easy to use
- facilitate collaboration between the different roles taking part in the lifecycle
- promote design, development, deployment and operations behaviour which supports reuse, flexibility, comprehensibility and visibility by providing guidance and structured assistance – making such behaviour the "path of least resistance".

Designing solutions

In our model the design phase comprises a wide range of activities – from requirements analysis through to service design and test specification, as shown in figure 3. A wide range of capabilities is required from tools – and in order to minimise the inertia which will inhibit designing for reuse, flexibility and so on, it is vital that access to common design and requirements artefacts is possible across the set of tools in use.

Figure 3: Roles, activities and capabilities in the *design* phase



In the design phase, architects help to balance conflicting requirements using knowledge of the business in order to maximise future reuse possibilities.

A key challenge in the design phase is to align the flexibility and reuse requirements of solutions with business needs; but to balance those against efficiency and openness requirements. The challenge of this balance in the SOA context is likely to be felt particularly keenly, because openness and flexibility are often considered to be "done deals" - but in general maximising these qualities increases the complexity of software – which of course impacts efficiency.

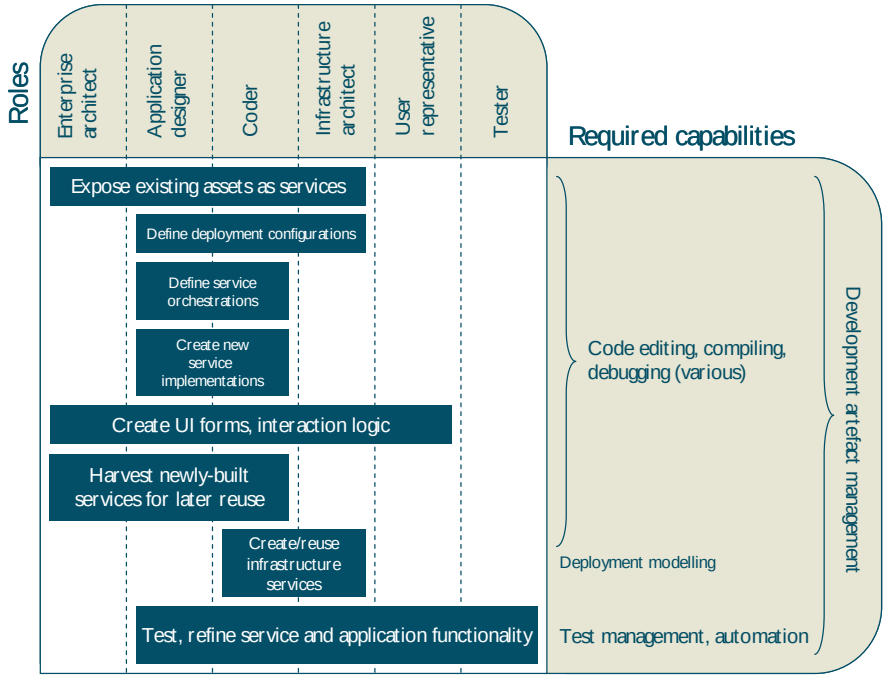
"Design for reuse" is a crucial element of this phase of the lifecycle if the benefits of SOA are to be realised, but it is a complicated process that cannot be applied blindly. Balancing these conflicting requirements can be achieved by an understanding of the business processes and priorities which are driving service design. "Efficiency first" will be the most worthwhile strategy in cases where the usage scenarios for supporting services are likely to be well-understood and stable. By contrast, "flexibility first" will make most sense where the service will be used over time in multiple contexts, and/or where there will be significant demand for change.

Requirements documentation and software modelling tools used in the design phase should encourage analysts, architects and designers to express these tradeoffs, along with other key quality-of-service and functional requirements, in well-formed service contract definitions which can form a consistent service and application design reference throughout the lifecycle.

Building solutions

As shown in figure 4, in our model the build phase comprises activities that look very similar to those found in "traditional" software build processes, with three important additions.

Figure 4: Roles, activities and capabilities in the *build* phase



Asset integration, deployment modelling and harvesting for reuse all increase the complexity of the build phase.

The first addition is the explicit inclusion of "asset integration". In the context of a SOA initiative, the line between development and integration is very blurred, so the application delivery lifecycle cannot consider integration as something that happens just before deployment – it is an integral (no pun intended) part of service development. Tools providing supporting capabilities in this phase must help developers, architects and managers understand the composite application as a whole, as well as letting them drill down into service implementation development and asset integration development activities.

The second addition is the creation or reuse of infrastructure services. Core to SOA is the idea of composite applications which are built on shared service foundations. If this approach is followed, the infrastructure underpinning those shared services must by definition also be shared – which in turn means that the benefits of SOA initiatives cannot be wholly realised unless you pursue application infrastructure investment and implementation as a "horizontal" platform proposition. Consequently the build phase needs deployment modelling tools which can help infrastructure architects discover the services provided by available infrastructure, and specify links between application

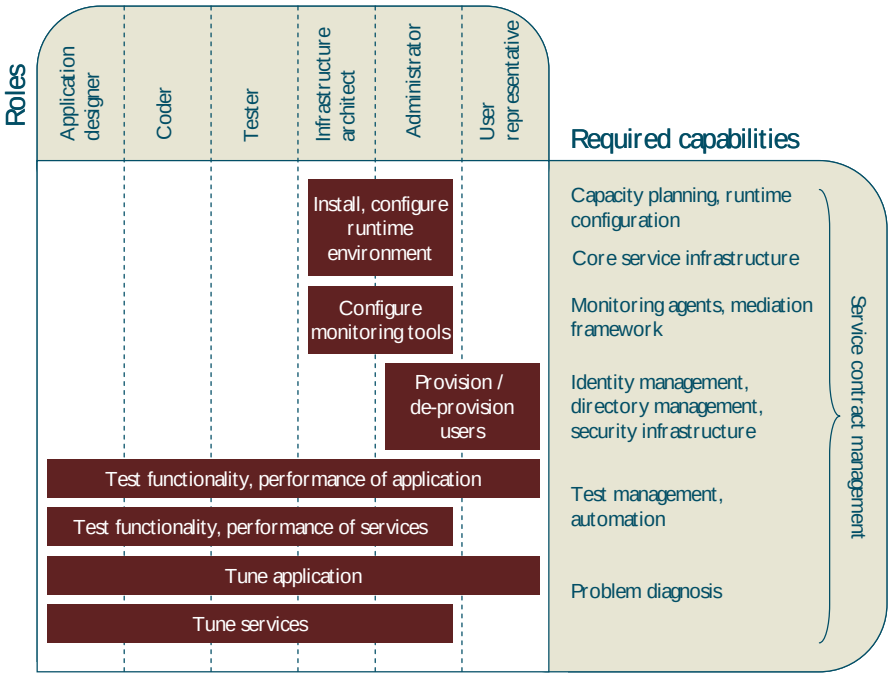
services and those infrastructure services.

The third addition is the harvesting of newly-built service definitions and implementations for later potential reuse. In order to maximise the quality of the build phase, the process of harvesting services for later reuse should use the relevant artefacts (code, interface descriptions, and so on) from the design phase as a key information source. This of course requires that artefact management capabilities from the design phase should be easily accessible from the development tools used within this phase.

Deploying solutions

Figure 5 outlines the deploy phase of our model.

Figure 5: Roles, activities and capabilities in the *deploy* phase



The specialised nature of the SOA runtime platform is the key issue in the deploy phase. Tools to help configure the platform and associated monitoring and diagnosis tools are crucial, as is the ready availability of information concerning service contracts from the design phase.

The specialised nature of SOA runtime platform elements is the primary differentiating factor in the deploy phase. Applications built within SOA initiatives require specialised runtime platforms – application servers which provide advanced facilities for publishing service interfaces and hosting their implementations; Enterprise Service Bus (ESB) implementations which can transparently apply quality-of-service policies, transformations and so on to service requests and replies through "mediations"; service orchestration engines; and so on.

The distributed, specialised, and shared/horizontal nature of SOA runtime infrastructure brings a challenge to the deploy phase. Developer and administrator productivity are easily compromised (and the overall ROI of SOA projects significantly weakened) if a great deal of effort is required to deploy services which have completed development and testing. Consequently the deploy phase will benefit a lot from tools which help infrastructure architects and administrators automate the packaging of code, deployment descriptors, policies, and other associated configuration information, and the remote installation of the resulting packages.

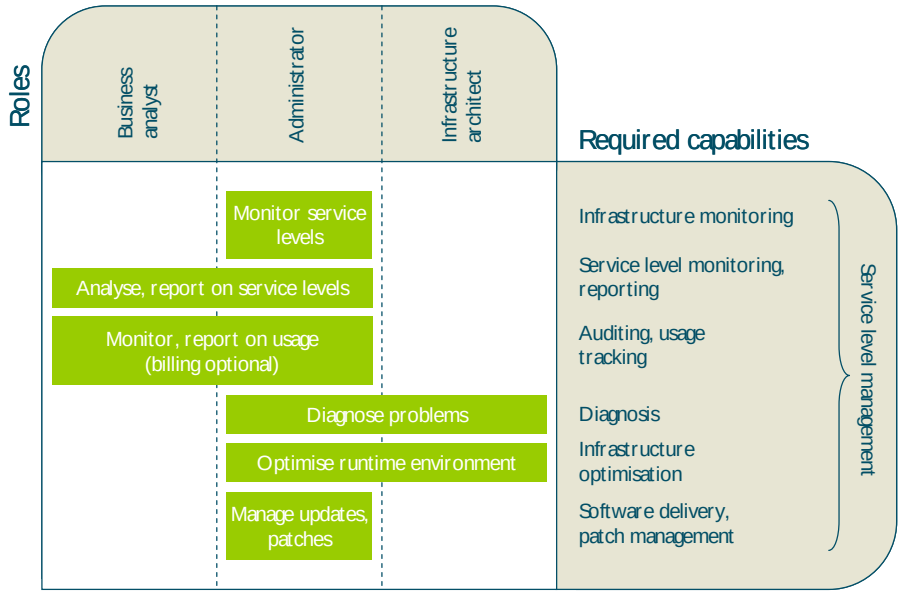
Another important factor which improves the productivity and quality of the deploy

phase is the ability of infrastructure architects and administrators to refer back to requirements documentation, design documents – and the service contract definitions which should be modelled within them – when installing and configuring runtime platform elements and the monitoring and problem diagnosis frameworks which need to plug into them.

Assuring the quality of solutions

Figure 6 outlines the assure phase of the lifecycle.

Figure 6: Roles, activities and capabilities in the *assure* phase



The ability to capture health and performance information at the level of services and inter-service information flows, as well as at the level of the encompassing composite applications – and the ability to analyse these in the context of established quality-of-service thresholds described in the design phase – are vital to achieving software comprehensibility and visibility.

As described earlier in this report, in SOA, the unit of software design – the service – also provides an appropriate and (if designed properly) business-meaningful unit of measurement at runtime. In addition, systems built from open service networks provide an environment where the tracking and logging of business-meaningful events is potentially straightforward. Taking advantage of these facts will improve the quality of the SOA lifecycle, and will also help to demonstrate the quality and value of SOA implementations to all stakeholders.

In order to do maximise the comprehensibility and visibility of software solutions, SOA initiatives will need to employ infrastructure software platforms which can monitor request and reply streams, and collect operational statistics about service response time and availability, and overall round-trip request processing. These need to be complemented by tools which store these monitoring statistics, and provide reporting and analysis facilities against the statistics and stored quality-of-service thresholds, and which can alert administrators and managers about potential or actual policy breaches.

Closing the loop: managing change

Mastery of version and change management is critical in any non-trivial SOA initiative. The nature of SOA is that it should be considered and pursued when change is to be not only expected, but encouraged. As groups of networked, shared services evolve and grow over time, and the promise of reuse materialises, individual services are likely to become dependent on each other to an ever greater extent. This increases the potential impact of change – and, more importantly, the complexity of assessing and managing its impact.

Moreover, as a SOA initiative becomes established and the IT delivery organisation improves its execution, multiple instances of the processes within each phase of the application delivery lifecycle are likely be running at any one time – with each instance probably utilising some assets (people as well as software) which are also used within other instances. This is difficult enough – but when you consider that solution requirements are likely to be evolving continually and driving change through multiple instances of the delivery lifecycle, it becomes clear that a razor-sharp focus on change management is paramount to avoid the risk of application quality degradation.

In order for the inevitable swirling vortex of change surrounding an ongoing SOA initiative to be managed effectively, change has to be pinpointed, analysed and enacted with reference to a "single point of truth" which gives change implementers reliable information about the demands and capabilities of other services which interacts with the service to be changed. The good news is that SOA's very nature can give us this information, in the form of service contract definitions which encompass not only key information about functional service interface specifications, but also information about non-functional behaviour – such as quality-of-service commitments, security requirements, and so on. Change management will be much easier if, as we describe above, contract definitions are made a primary output of the design phase, and then made a primary reference point for staff playing roles in the build, deploy and assure phases.

Any broad SOA governance framework you put in place therefore needs to mandate a lifecycle methodology that doesn't just provide a clear structure for directing design and development strategies towards reuse, flexibility, comprehensibility and visibility of applications and the services within them. It must also drive the people playing key roles in the application delivery lifecycle to manage change effectively through rigorous application of, and reference to, service contract designs.