

Your SOA needs a Business Case

Piet Jan Baarda
Senior Information Architect
Sogeti
pietjan.baarda@sogeti.nl
Version 1.1 ENGLISH – November 6, 2008

Over the last couple of years there has been a tremendous interest in Service-Oriented Architecture (SOA). It can safely be classified as a hype. As hypes go it follows the Gartner hype cycle. We now seem to be in 'Trough of disillusionment'. Signals of 'SOA fatigue' are appearing everywhere. Professional architects, of course, are not concerned with hypes, and have recognized the true value of SOA from the start. This article will illustrate SOA's true value by describing practical scenario's where cases exist for developing services. The actual composition of a complete business case will not be described. Aspects specific for service business cases are treated.

Introduction

Recent research shows that a only small percentage of projects executed with a SOA label actually deliver the expected business benefits¹. Just like many other hypes SOA is based on a good idea that has been hijacked and changed beyond recognition by many vendors in the marketplace.

On one hand some software suppliers will lead you to believe that the acquisition of an Enterprise Service Bus or the development of a number of Web Services is sufficient to be able to say 'Yes, we do SOA!'. Many customers happily join them and count on the next miracle tool to make all their information problems go away without the need to do complex things like changing the organization or the current way of working.

On the other hand there are those that *do* recognize the need to change the organization and the way of working. But they then tend to go all out and believe that services should be applied across the board as the cure-all paradigm.

This article will show the sweet spot is, of course, somewhere in the middle. There is no miracle tool and blind implementation of services is not a good idea. In practice there are some scenarios where they are worth the effort. And there are other scenarios where they are not.

The good idea at the basis of SOA is – more than many other hypes - based on existing and widely accepted concepts. It concerns concepts like componentization, contract-first, reuse, standardization and, last but not least, business focus. Although these concepts are accepted and understood, the actual application is not easy, as indicated by the fact that they are still not normal practice for most organizations. Whether they are part of a SOA initiative or not. Many organizations are not able to

¹ Illustrative is a study by Burton Group (June 2008). It shows that 50% of the investigated 20 companies classify their SOA initiatives as a complete failure. 30% classifies it as neither successful or failure. Only 20% of the companies was prepared to call their SOA initiatives successful.

create the enterprise architecture and governance competence that is needed to achieve the benefits of the application of these concepts. They continue to execute autonomous projects.

This article may help to improve this by providing instruments to give focus to SOA initiatives and deflate the hype factor.

The premise is:

“To develop a service there must be a clear business case”.

We will discuss practical matters, suggestions and considerations to support the above premise. Practical guidance will be provided for those wanting to apply the power of SOA in their organizations. Most statements are in line with a vision of SOA as applied and expressed by Sogeti. This vision can be found in the book ‘SOA for Profit’². It is also aligned with research into ‘Enterprise Ontology Based Service Portfolio Management Methodology’³, performed by the author at the Delft University of Technology.

Many definitions for Service-Oriented Architecture exist. In order to get the message across it is important to clarify what our definitions of the key concepts are. Or rather what we see as the true power of SOA is or what it should be. Then we will describe typical scenario’s where a business case exists for developing services.

Service-Oriented Architecture

There is no need to discuss all existing definitions of SOA or formulate a new definition of our own. It is sufficient to mention the core of Service-Oriented Architecture that all these definitions agree upon.

“Service-Oriented Architecture is an architecture style that enables the improvement of enterprise agility through the use of business services”.

To present a really clear picture we do need to investigate the following concepts more closely: *architecture, enterprise, agility* and *business service*. Also a few remarks will be made on *business service portfolio’s*.

Architecture

SOA is an ‘enterprise architecture style’. Many examples of failed SOA initiatives can be traced back to shortcomings in the architecture- and governance competence of the enterprise. The old way of working is used with autonomous projects creating point solutions with no regard for enterprise consistency, coherence, conciseness or comprehensiveness. In these failed cases, people are basically extending the existing silos with services.

² Martin van den Berg, Norbert Bieberstein, Erik van Ommeren; ISBN 978-90-75414-14-1; May 2007

³ The research hypothesis is that a business service portfolio based on a model of the implementation independent essence contributes most to the promise of SOA, the optimization of the enterprise agility. Such a model can be created using the DEMO methodology. DEMO is developed by Prof. Dr. ir. Jan L.G. Dietz at the Technical University Delft in the Netherlands. Sogeti in The Netherlands uses PRONTO, a DEMO derived approach, for their process management engagements.

Agility will only improve when a coherent set of business services is created that can be used in many orchestrations without a need to perform cumbersome transformations to translate information concepts (especially semantics!) from one silo to those of another. The coherence will only be created with a sufficiently mature architecture- and governance competence. The determination of the effort needed to ensure this maturity is a very important part of the cost-benefit analysis for a business case. This effort is often underestimated or even ignored.

Agility

Agility in the SOA context is defined as the ability of an enterprise to thrive in a continuously changing and unpredictable environment. This is not the same as flexibility. Flexibility in a business setting is defined as 'the ability to move from one task to another with each situation defined ahead of time'. Flexibility is a prerequisite for agility. In order to be agile an enterprise must be flexible, and in addition be prepared for an unpredictable future. Other SOA benefits that are often mentioned are all contributing to increasing agility. Examples are reduction of integration cost, increase of reuse, reduction of risk, reduction of impact of new regulations, etc.

Enterprise

Enterprise can mean many things, the essence is captured in this formal definition⁴ *“a consciously coordinated social entity, with a relatively identifiable boundary, that functions on a relatively continuous basis to achieve a common goal, or set of goals”*.

It doesn't matter if it is one part of an organization or a complete value chain, as long as we recognize we're trying to achieve some common goals, that there is some boundary and, most importantly, *within* that boundary we have *coordination* and (hence) *integration*. And to make the whole more agile, we need to examine the inner workings of the 'enterprise'

Business services

In SOA literature many different service-classification schemes can be found. In our vision it is all about *business services*, services that have a meaning in the business domain, can directly be used for the implementation of a business process and are an IT artifact as well. The consumers of the business services are applications, BPM tools or other business services.

There are also other service types, by some called utility or technical services, that belong to the IT domain and are there 'just' to implement business services. The efficient implementation is an IT problem, and is not specific to SOA. Many SOA concepts can be applied in the IT domain also. They are not seen as part of the services business cases we are talking about here. A business service is treated as black box with a fully defined interface, the 'what' of the service. This includes quality of service and conditions. The 'how' part of a service is not of interest for the consumer (application, BPM tool or other service) and can be changed without any changes for these consumers. The 'how' contains business logic as well as technology, and is adapted as needed by the changes in the environment. The 'how' determines, of course, to a large extent the cost and feasibility chapters of a business case.

⁴ Robbins, S.P.: Organization Theory. Prentice-Hall, Englewood Cliffs (1990)

Business Service Portfolio's

For optimal agility the portfolio is *coherent, consistent, concise* and *comprehensive*.

- Coherent means the portfolio forms a logical and integral whole. Coherence is determined to a large extent by semantic interoperability. If interoperability has boundaries along organizational or even system lines, agility is seriously compromised. The service consumer is confronted with (semantic) transformation issues.
- Consistent means the portfolio is free from contradictions and irregularities. Agility will suffer if the portfolio is inconsistent as the inconsistencies are to be removed by the consumer.
- Concise means there are no superfluous services in the portfolio. Otherwise there are more services to maintain and govern and the portfolio is less transparent for consumers.
- Comprehensive means the complete scenario is covered. If not, agility is lower as the missing services have to be added when needed. Introducing services 'just in time', when they are needed is a good approach, but until the needed services are present agility is sub optimal.

One approach to developing a service portfolio that is sometimes promoted is 'survival of the fittest' where services are defined without regard for the above properties and letting evolution decide how the portfolio will develop. The assumption is that in the long run the properties will be met through natural selection. There is no evidence that this will actually happen though.

It is safe to assume that this will happen only by design, through architecture and governance and not through chance.

If architecture or governance is lacking, just of bunch of services (JBOS) will be created with no improvement on enterprise agility. Architecture defines the principles and standards to use for defining and maintaining the service contracts. Governance makes sure these are actually applied through service portfolio management and making sure the services are used where appropriate.

Business Case Scenario's

In a situation with a continuously changing and unpredictable environment a case can often be made for business services. This is the promise of SOA, to allow companies to thrive in such an environment. The challenge here is to limit the impact of any changes to the internals of services and keep the service interfaces stable.

As SOA concerns the use of *business services* to achieve enterprise agility, the business case concerns the introduction of a *portfolio of business services* and the associated cost-benefit analysis.

Good business cases for developing business services can often be found in one of the following scenario's. The business benefits are clear and are often essential for the survival of the enterprise. If feasibility is an issue it is often attributable to a very fragmented IT landscape that is a burden to the enterprise in any case. In *all* the scenario's the environment is continuously changing, on the business

side, and often also on the IT side (see Figure 1). SOA has benefits, by definition, only where agility is required⁵.

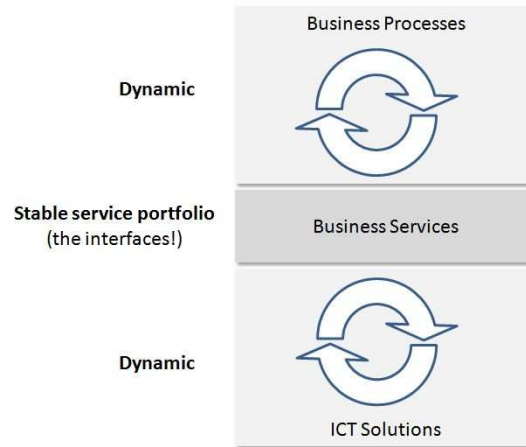


Figure 1 - Dynamics and business services

A business case follows from research into the expected dynamics in six different areas or combinations thereof (see also Figure 2 on page 9).

1. Products and services
2. Regulation
3. Channels
4. Acquisitions
5. Hosting
6. Business to business

The business benefits will be determined together with cost and feasibility in general and feasibility of the needed architecture and governance competence in particular.

The business scenario's are described below:

1. **Products and services⁶**. An enterprise in this scenario needs to adjust continuously to changing markets. Adjusting in time is a matter of survival. Traditionally new systems were built for each new product or service. With increasing dynamics and a faster rate of writing off investments this cost per system is increasing. Typical examples are companies with extremely short product lifecycles as for example found in Telecom, Consumer electronics and technology in general. The argument for using SOA is breaking this cycle and minimizing the impact of new products

⁵ Of course there may be a good business case for rationalizing systems with benefits like cost reduction on hardware, maintenance, license cost as well as improving business agility. This is not the type of business case we are talking about here. We are talking about business cases where **business agility is improved through the use of business services**.

⁶ Please note that the word services as used here (as a product) is something different than the SOA business service. The first is dynamic while in SOA we strive for stable service interfaces.

through reuse of services. Part of the case is also the lowering of the threshold to introduce new products and seize market opportunities.

2. **Regulation.** This is very close to the previous scenario. Quite possibly the unpredictability is even greater. Examples can be found in organizations involved in areas with ever changing laws. Tax collection and social security-agencies come to mind. But also organizations in health care often have a business case for services to minimize the impact of new regulation. And these days even financial institutions.
3. **Channels.** Many organizations use multiple channels to offer their services to customers. For example banks, insurance companies, travel agencies and air lines. Each channel may have functionality for their specific audience. But a customer may interact with the organization through multiple channels. They often implement the 'same' business process and need to ensure consistency across channels. They should be based on the same set, channel independent, business services. That makes it possible to quickly implement new channels. Only adding elements specific for the channel. The dynamics come from channels that come and go. This scenario very often occurs. Especially web frontends have a very short life span. Most organizations already have multiple channels in use. The trend is further segmentation and more channels. The cost per channel when using reusable services very quickly becomes much lower than the alternative approach, creating new systems on the underlying legacy systems for each new channel. Still, this is still common practice because a limited maturity makes it is much easier for organizations to manage autonomous projects.
4. **Acquisitions.** This scenario includes mergers and takeovers as well as the splitting of organizations. In all cases systems must be integrated or separated. When applied correctly services allow this to be done with transparency for the business processes. The integration or separation takes place *behind* the service interfaces. In time, the backend systems may be rationalized without impact on the business processes. There can be a fixed set of services that are changed *internally* to integrate new systems, without any changes to the service interfaces. Often financial reporting has priority in this scenario and can be implemented using a fixed set of business services. An example is a company in the oil business that has a habit of taking of and selling on average a dozen subsidiaries a year. Sometimes systems are integrated by simply replacing them with standard software. Some banks use the this approach. In fact it is a way to quickly implement the required services. The dynamics of this scenario are formed by systems that need to be integrated, uncoupled or replaced. The business case consists of the advantages of a lower threshold for acquisitions and the minimal disturbance of business processes. The cost of the alternative, custom point-to-point integration, must be outweighed by the cost for the development of the architecture and governance competence and any infrastructure that is needed for services. When only one or a few acquisitions are envisioned the case is obviously weaker than when the future survival of the organization is dependent on frequent acquisitions.
5. **Hosting.** A good example of 'hosting' is the reservation system of an airline that is made available to other airlines and travel agencies. But also a bank doing 'white labeling' is a fine example. Instead of implementing a customized instance of an application for each party to be hosted a common set of services is created for the common denominator. Processes (service orchestrations) can then be specific for each hosted party. To be clear: with service we mean the

service *interface*. The logic within a service may contain parts specific for a hosted party and may need changes caused by changing circumstances. The dynamics consist of coming and going hosted parties and the changing business processes of those parties. The business case depends on the market for hosting and how this fits in the strategy of the organization. A comparison is needed with the alternative, implementing, customizing and maintaining separate instances of a system. This is independent of the deployment of the application. It may be installed at the site of the hosted party. When this is not needed and all interaction takes place through the internet, this is called 'Software as a Service' (SaaS). This scenario concerns the organization offering hosting services.

6. **Business to business.** The original vision of SOA included a global Yellow Pages directory containing electronic services that fit seamless and could be used in an integrated way and even be discovered automatically. In practice this proved to be a 'bridge too far'. Currently we are not yet able to describe abstract interfaces in such a way that they allow fully automated negotiations over the semantics, service levels etc.. Successes are found in the following domains:

- Travel organizations where a complete trip transaction can be performed including flights, hotels and cars in an integrated fashion.
- Banks with payment transactions on each other's accounts (SWIFT).
- Service aggregators, making it possible for a customer to maintain all his accounts, even when they are serviced by multiple banks. Large companies never do business with one bank, but limit the share of wallet for one bank. As each bank is only able to provide an incomplete view the customer is responsible for his own overall cash management or can make use of an integrator.

By defining the services in such a way that many parties can be accommodated, great flexibility is created with opportunities to quickly seize business opportunities. The dynamics consists of new parties and new products. The enterprise is the chain of businesses. The business case often involves the survival of the enterprise as these business-to-business interaction is tied to the nature of the business and the expected developments. The challenge is to align all involved business and create the right service portfolio.

Sometimes there is no alternative but starting a separate organization. Although started long before the term SOA was coined a very good example is SWIFT. SWIFT is an organization founded in 1974 by financial institutions in order to process financial transaction in a standard way. Nowadays it is being used by 8.200 financial institutions. Aside from the formulation of a standard 'language' for communication on financial transactions (ISO20022), there is also a technical platform. In general the biggest challenge in this scenario is making sure the 'enterprise' truly cooperates and a coherent set of business services is created.

A business case is not always self evident here. Some forms of cooperation can be very threatening. Aggregators are often seen as snakes instead of true partners.

7. **Combinations.** In many organizations cases exist that consist of more than one of the above business cases. A good example is tracking & tracing in logistics companies like mail handlers. A case of a combination of the business-to-business and channel scenario's. There may be

separate cases, so each case can be developed profitably separate from the other. Still it is advised to strive for a coherent set of services across the scenario's. The effort needed is mostly not larger than the sum of the separate autonomous development. The result is greater agility in unforeseen future situations. Understand the relative concept of 'enterprise'. A combination can consist of case for a chain of companies together with a case for a department.

Business cases

The business case concerns the *introduction of business services to achieve enterprise agility* in one or more of the scenario's described previously. A very natural concept to base the case on is the Key Agility Indicator (KAI). For example in the 'channel' scenario a KAI may be the time it takes to create a new channel. In other scenarios obvious choices are the time or amount it costs to introduce a new product, to implement new regulations, or integrate a new acquisition. The benefits are expressed in terms of improvements in the KAI's together with the business value of these improvements.

The diagram below (Figure 2) illustrates the discussion above. It may seem like a mathematical approach, but that is clearly not the case. To determine to what extent certain dynamics justify investments in business services and what is needed in organizational and competence terms is rarely a clear 'yes' and 'no' matter.

There is NO case for pure technical goals. This is so because SOA is, by definition, about achieving *business agility through the use of business services*. So a SOA business case must describe the benefits in those terms not in terms of technical goals. So SOA is not an answer to integration problems. The goal is the improvement of business agility through the use of business services. The solving of integration problems *is* a prerequisite for the realization of services and determines cost to a large extent. The next paragraph will also address this issue.

It is clear many choices have to be made before a business case can be implemented. Are the part-business-cases developed separately or as a whole? What are the demands for the architecture and governance competencies? When there is no business case for certain parts, how do we recognize a future situation where there *is* a case?

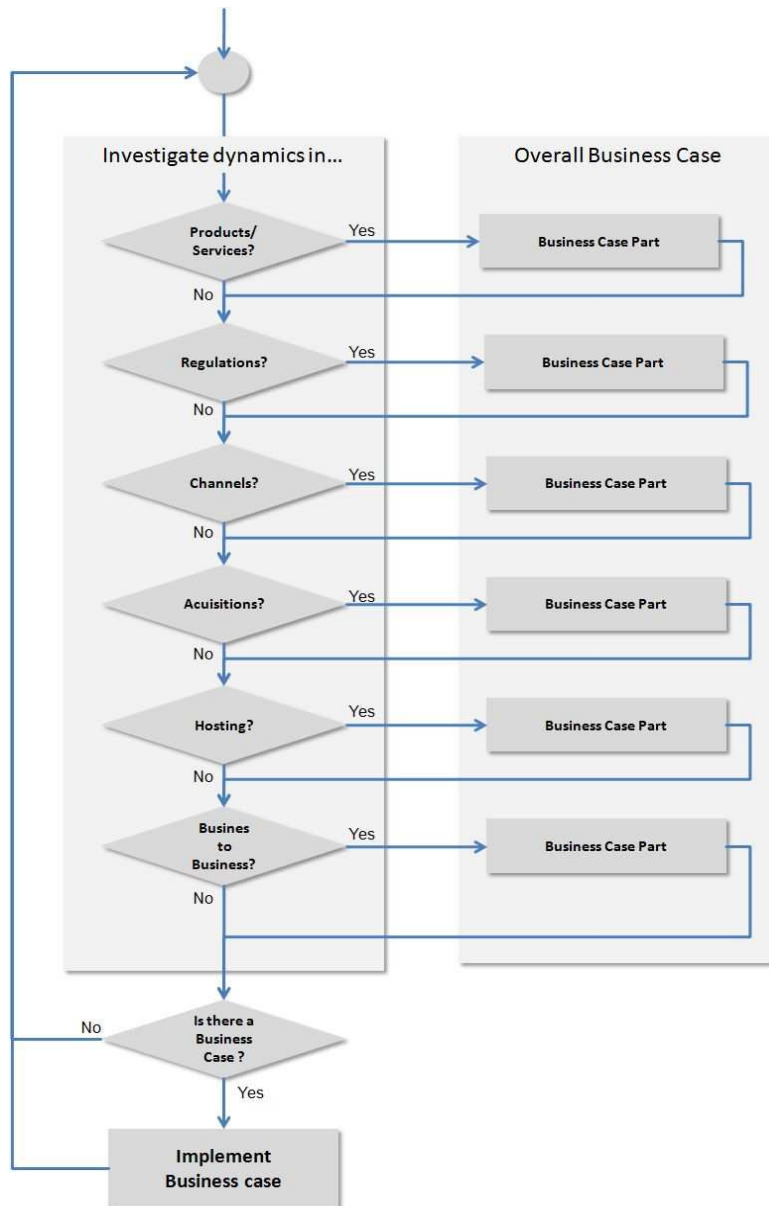


Figure 2 - Dynamics and service business case

A crucial element of the business case is ensuring the enterprise architecture competence is in place, together with governance en top management support. Otherwise the agility improvement that is the goal will not be achieved and will maybe even negatively impacted. Service portfolio management must be centralized as a coherent set of services will not appear through autonomous projects and projects have to be steered towards using the right existing services and preventing inventing alternative solutions. If it is not feasible to achieve sufficient architecture and governance maturity, the agility case is unfeasible also. The organization should work on developing this instead. Organizations tend to overestimate their own maturity levels. That this is not trivial is proven by many failures in the category (tax, defense, police and immigration).

When *no* services?

Looking at the preceding scenario's, this question is easy to answer: Do not develop services in situations where none of the scenario's apply.

The following properties apply all:

- Static products and services
- Static regulations
- No plans for acquisitions
- No hosting
- No channels that come and go
- No changing partners to exchange information with

Or...

- Unfeasible. The cost is too high, the performance requirements cannot be met or the architecture and governance competence is not present or cannot be developed.

In particular companies producing a very long running product or product line, where no changes are expected and sell through wholesale intermediates where also no changes are expected. A strong indication of a case where no services are needed are those where there is little spent on integration in the current situation, without SOA. When a large chunk of the IT budget *is* spent on integration, there is a big chance there is a case for services. As said before, there is also no case when the demands for mature architecture and governance competencies cannot be met.

It may come as a surprise that from these scenario's no case can be formulated for the application of SOA to solve fragmentation-, legacy- or silo-problems. If the quality of information and IT in general is too low and its cost is too high there is probably a business case to solve this. But, this is not a SOA business case as the solution is found in the rationalization of existing systems and not in the development of business services. Again, a SOA business case is by definition about increasing business agility through the use of business services.

An argument that is sometimes used against the use of services is the difficulty of meeting performance requirements. This argument mainly follows from mixing up the SOA concepts and specific technology choices. Performance is often an issue when complex multilevel composite Web Services are used, possibly with several backend systems with the associated protocol, syntax and semantic transformations. But often it is worthwhile to look beyond a solution with Web Services or an ESB. Being technology agnostic is attractive, but should not be a show stopper. Also if service composition makes a service too slow, always a non-composite implementation of that same service is possible. Ultimately if performance requirements really cannot be met the service should be classified as unfeasible.

Paradoxically even in situations where there is no case for services, still the SOA paradigm is applied. When no case is found it is pertinent to keep applying the SOA paradigm and keep a sharp eye for changing circumstances. There may be a time when a case for services does arise. This way services can be introduced 'just in time'. This applies to any *enterprise*: a complete chain of companies, a single organization or a department within a company.

Conclusion

In 'SOA for Profit' it's already mentioned: "A certain way to get fired is to propose to '*migrate everything to SOA*'". In the book it is made clear that there *must* be a case. This statement is further investigated and particular scenario's are described where these case are often found. The following scenario's are found:

1. Products and services
2. Regulation
3. Channels
4. Acquisitions
5. Hosting
6. Business to business
7. Combinations of the above

When no such case is found SOA is still applied as an architecture style. It was used to determine there is no case in the first place and needs to be applied continuously to recognizes changing circumstances. It allows you to tackle opportunities just in time. Without SOA great opportunities may be missed.

A business case for rationalization of backend systems is not a SOA case.

When you are thinking about the application of SOA in your organization, or have already started, ask yourself these questions:

- Is there indeed a need for agility?
- What are the concrete benefits of each service?
- Do the benefits outweigh the costs?
- And... is the architecture and governance competence sufficient to ensure the quality of the service portfolio needed to achieve the expected agility?

Many different sources have been used for this article; books, articles, blogs and research into the application of SOA. The latter category includes research performed by Oracle that formed an important inspiration ('Bringing SOA Value Patterns to Life, An Oracle White Paper', June 2006). In this report several application categories were recognized. The main arguments come from experience in the field gained at customer SOA engagements. This article has no scientific aspirations and no bibliography is included.