

Maconomy White Paper



Service Oriented Architecture for Small and Medium Sized Businesses

MACONOMY

–people made profitable

Summary

Many small and medium sized businesses have not embraced the idea of a Service Oriented Architecture. This paper argues that SOA presents great opportunities for SMBs but that the implementation (a) Should be incremental, (b) Should focus on integration and business processes, and (c) Should preferably be based on open standards. This way SMBs can enjoy the benefits of a flexible and adaptable IT architecture supporting the ever changing business needs of the company.

Maconomy is 100% SOA enabled and in combination with Maconomy's best practice processes this gives the SOA architect the opportunity to easily integrate to other systems and create powerful intra-system business processes around the Maconomy system.

1. Introduction

Service oriented architecture (SOA) has for some time attracted the interest of system architects by promising a closer integration of the enterprise business systems, while at the same time enabling increased adaptability and agility. Forrester reports that 53% of all European and North American enterprises¹ are using or plan to use SOA within the next 12 months [1].

While the vision is very appealing, many companies have not yet embraced SOA. This is particularly true for small and medium sized businesses (SMBs) – the hype around SOA might blur the vision, and (mis-)lead SMBs to believe that implementing SOA is a large and expensive endeavour for which their resources could prove inadequate. Indeed, Forrester's research shows that while 67% of companies with more than 40.000 employees are implementing SOA at the end of 2006, the percentage for SMBs² is far smaller (44% report that implementing SOA is a high or critical priority) [2].³

The present white paper discusses why it is important for SMBs to have a clear strategy on SOA and how SOA can achieve advantages on the short term while gradually expanding the SOA.

Section 2 gives a short overview of the SOA vision, and section 3 discusses the challenges and opportunities that current state-of-the-art SOA poses to SMBs and makes recommendations as to how SMBs should face this challenge. Finally, section 4 explains how Maconomy will fit into any Service Oriented Architecture no matter if you are adopting SOA patterns in a case-by-case manner or whether you pursue the entire vision.

2. The SOA Vision

SOA addresses a number of issues well-known to many IT architects. In particular it promises to:

- Replace big bang investments with more iterative and adaptive development processes
- See technology as a catalyst for improving business processes rather than a constraint
- Enable reuse of components across systems and technologies
- Enable more flexible and responsive business processes

This chapter gives an overview of the technology supporting this promise

1 Forrester defines an enterprise to be a company with more than 1000 employees.

2 Forrester defines SMB to be companies with less than 1000 employees.

3 The numbers are open for debate: In their survey of IT decision-makers Butler Group found that only 8% had actually deployed SOA in a live environment with an additional 17% engaged in trials and 36% in the process of evaluating the process. Datamonit found that 30% were deploying or trialing SOA, and notes that adoption rates are highest in Technology and Financial Service verticals [6].

2.1 SOA Overview

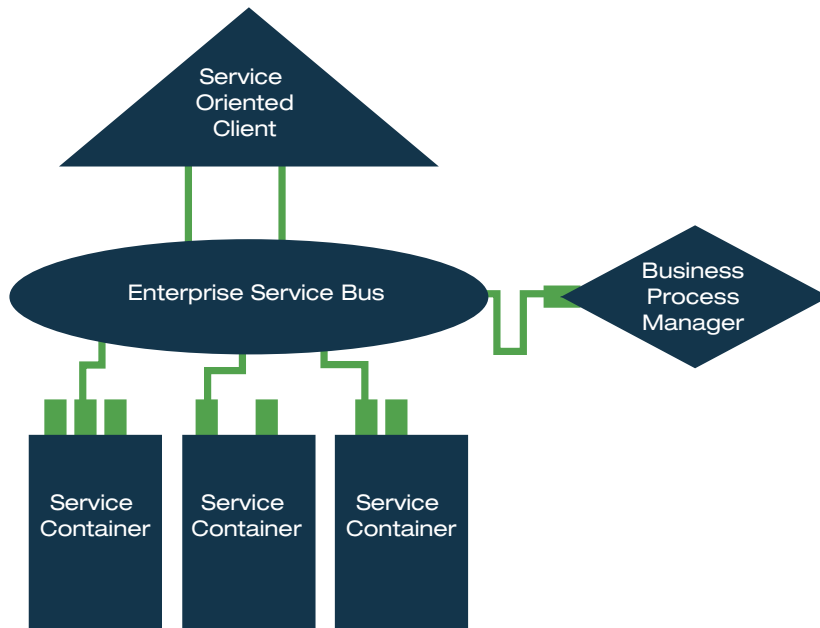


Figure 1

Figure 1 depicts an idealised, service oriented architecture. The basic idea is that independent service containers provide services, which makes sense from a business oriented point of view. I.e. the services offered do not expose the underlying technology or implementation. The idea of using services for interfacing to components of an IT infrastructure is not new – Corba, COM and EJB are examples of technologies that essentially tried to achieve this goal. These technologies failed to become widely adopted because they were proprietary and/or fixed to a certain platform.

A standard for providing services will enable architects to easily integrate diverse business systems and to reuse components. It is important to understand that facilitating integration is really the essence of SOA – no more, no less (IT architects will tell you that this is indeed important – also recall that integration need not be within your business but can also be b2b or b2c integration with vendors and customers). Nevertheless, the standardisation around SOA has enabled a number of off-spin technologies. This paper will address some of the most important ones as these are often seen as part of the SOA landscape. Spin-off technologies include:

- Business Process Management (BPM): Having a standardised interface to business-level services enables general Business Process Managers to orchestrate business processes across many business systems. The BPM might itself be seen as a service container.
- Service Oriented Client (SOC): Having a standardised client architecture allows the end-user to access the entire IT infrastructure using only a single user interface.



2.2 SOA Overview

SOA is not a technology – SOA is a way of organising, maintaining and developing your IT architecture. Nevertheless, a huge community of software vendors have arisen, providing software for supporting a Service Oriented Architecture.

2.2.1 Web Services

Web Services stand as the central technology of SOA. While SOA does not actually prescribe Web Services, the two concepts have become almost synonymous. While earlier attempts at SOA failed (see section 2.1) Web Services provided an independent, simple, public protocol for communicating, which has proven successful and which is now supported by all major IT vendors.

Web Services is a standard (or rather a collection of standards) for high level communication typically via internet protocols such as HTTP or HTTPS. One of the keys to the success of Web Services is the fact that it uses XML, which is an open document standard that is readable by human beings while being easy to process by computers. One of the central standards of Web Services is SOAP which specifies the protocol for querying a web service and receiving a result. SOAP is based on XML (a complete introduction to SOAP and other of the standards mentioned in this section can be found in [3]).

Communicating using Web Services alone does not create a Service Oriented Architecture. Each service container should publish its interface as coarse grained Web Services. This means that the entities handled by the Web Services are business level objects and similarly the services provided are business transactions (in contrast to e.g. database objects and database updates). This ensures a modularisation where components of the architecture need not have detailed knowledge of other components, thereby facilitating reusability and adaptability – the key promises of SOA.

While a Service Oriented Architecture in principle could be based around a number of components calling each other using web services, this would require that components knew where other components could be found, how they were called exactly, how they handled security etc.

Such issues are dealt with using a number of additional Web Services standards: UDDI can be thought of as the phone book of Web Services providing a central place to look up other web services, WSDL provides a standard for describing how to interact with a Web Service, a number of standards exist for handling security and authentication. Many vendors package technology implementing these standards as an Enterprise Service Bus (ESB). Whenever a component needs to invoke a web service, the ESB will locate the service, check the call and ensure that parameters are packaged with sufficient authentication tokens. In addition most ESBs will provide tools for transaction control, migrating services, monitoring your SOA, load balancing and much more (see [4] for Forrester's list of requirements for ESBs).

2.2.2 Service Oriented Client

There is no doubt that introducing one user interface embracing all business processes would make a huge difference to end users. Not only will the general user experience improve greatly, thus freeing resources otherwise tied up in awkward processes involving separate systems.

Perhaps even more important, such a client would allow users to have an integrated view of your business enabling them to make better decisions for the good of the company.

The technologies that try to meet this promise fall into two camps:

- Portal frameworks (such as IBM WebSphere or Microsoft SharePoint) which allow programmers to create web components accessing different business systems. In a Service Oriented Architecture web components will naturally access web services.⁴
- Generic Rich Clients (such as Sun NetBeans or Eclipse RCP) that provide the infrastructure for programming a user interface.

The portal frameworks have had considerable success over the last years as an integration platform. Web components are supplied out of the box by some vendors of business systems, and if not, they can be developed at relatively low cost, provided that adequate interfaces exist. In particular, if the business system provides a rich interface based on web-services, the .Net or Java based tools provided by Microsoft or IBM makes it relatively easy to construct a web component. The development cost of a whole portal accessing all business functionality of all business systems will, however, be huge, and require substantial skills in both technology, integration and usability as well as a deep understanding of the business processes involved.

An additional reason why few companies try to integrate all business functionality into a portal is the fact that the browser based interface on which portals are based is simply not adequate for all business processes. While modern AJAX technology has improved the responsiveness of the user interface, it will never reach the levels of usability and quick interaction of a rich client. Hence, portals might be a preferred solution for providing a common gateway to a set of lightweight interactions for some users, and it is able to collect and present business data from diverse business systems to decision makers and data analysts. But it will not be the preferred user interface for the users that use one or more of the systems as a main tool for working, and even more infrequent users might find the usability inadequate.

The best alternative to a web-based client is a rich client⁵. In recent years, frameworks have emerged promising to take the toll out of writing a client from scratch. The idea is that all clients have a common skeleton that can be written once and for all (this could include server communication, handling menus, frames etc.). The current offerings do provide the basic infrastructure of a client but still leave a lot of programming to the developer of the client. Both NetBeans and Eclipse are very successful, integrated development environments (IDE) and are therefore of a sufficiently general nature to supply both the functionality of an IDE and the functionality of a business system. This generality, however, has the penalty of requiring more work to develop a client. So while the cost of developing a complete web portal was huge, the cost of developing a full rich client will be even bigger.

The bottom line is that current technology is not yet sufficiently mature to deliver a service oriented client covering all business processes – portal frameworks are suited to present data from multiple sources and to integrate processes where responsiveness is not a key concern. Generic rich clients address

4 Note, that there are no standard definitions of a Service Oriented Client – some authors use the term as an alternative to browser based interfaces. Furthermore, you often find additional requirements such as ease of deployment or the richness of the user interface. Here we just see the SOC as an integrated user interface to the functionality and process of a SOA.

5 A rich client is basically a fat client which is as easy to deploy and upgrade as a thin client.

the problem of responsiveness, but due to their inherent complexity, it is not a realistic task to create a rich client covering all processes. Instead, important, much-used, intra-system processes should be the focus of introducing a generic rich client.

2.2.3 Business Process Management

In a web service based SOA, a business process is itself a web service which is defined by composing other web-services. The most widely known standard for orchestrating business processes is known as BPEL (Business Process Execution Language). BPEL is a standardised (OASIS) XML language specifying the functional aspects of business processes such as flow of control, synchronous and asynchronous coordination etc. Hence, BPEL can combine web-services such that business processes involving activities in many different systems are automatically executed. A transaction in an ERP system might initiate a BPEL process which in turn starts transactions in CRM and Inventory systems. The BPEL process decouples the systems completely – the ERP system does not even need to know which systems are affected by the process (you might add a wage system to the process by simply changing the BPEL specification). A BPEL process might also initiate transactions that involve web services defined at a customer or vendor site even when not all details are known about the internal workings of such sub-processes.

BPEL has been designed with the aim of automating business processes. Business processes that require human intervention (such as the approval of a vendor invoice), are not supported directly. One reason for this is that human intervention is closely linked to the business logic of the individual sub systems (such as the approval hierarchies of users) and alerting a user of the need to intervene must be part of the user interface. These problems can certainly be overcome, but such solutions will easily result in a tight coupling of web services which in turn will limit the generic benefits of the SOA. Many SOA vendors provide extensions of BPEL beyond the standard – the cost of using these is vendor lock-in.

3. SOA for SMBs

Section 2 reviewed the benefits of introducing an SOA. A couple of warnings were also given, particularly against standards that are not yet fully finished (or vendors not complying with them) and in the case of SOC against the huge development costs entailed if the SOC should cover all business functionality. This section will look at the benefits and problems with SOA as seen from the point of view of a small or medium sized business.

3.1 Using SOA in SMBs

While there is certainly great diversity amongst SMBs, the following properties are typical:

- An SMB has a small IT department and rarely has the skills and resources to take on large development projects.
- An SMB has a manageable number of business systems. The IT department has a fairly good overview of which ones are in use. Often one business system plays a central role, while a number of supplementary systems are employed for specialised purposes.

SOAs promise of easing integration of business systems is as relevant for SMBs as it is for large enterprises – having fewer systems does not make integration less important or less painful. Adopting an SOA is perhaps even more important to an SMB than to a large enterprise: because of their limited IT resources, SMBs should look towards proven practices and if possible avoid vendor lock-in by choosing open standards. As we have seen, the combination of web services and a loosely coupled structure can deliver this. In other words, the first step of an SOA strategy can be simply adopting SOA as a pattern for system-to-system integration. This can even be done incrementally and provides a very cost efficient strategy for introducing SOA.

Introducing SOA as an integration pattern will come a long way in reducing the cost of integration and will detangle some of the complexity of tightly integrated systems. For business systems of a certain complexity, the simple approach still misses some opportunities for simplifying the integration and maintenance of complex business systems. In particular, each consumer of a web service will need to have detailed information on the expectations of the web service in terms of call pattern, security, location etc. The next natural step in an SOA strategy will then be the introduction of an ESB.

It is important to realise that the introduction of an SOA can indeed be incremental as described above, and much of the work done to make direct integrations between business systems will still be useful when introducing an ESB (if the intention from the beginning is to introduce an ESB, there is no point in postponing the introduction of this infrastructure).

The actual need of the business must be examined carefully before choosing an ESB. Many vendors provide fully integrated ESB with extensive functionality, which is well beyond the scope of this paper. These ESBs often go hand in hand with consultancy and are offered as a package including best practice on the use of the ESB. As an alternative, open source ESBs are emerging (e.g. JBoss, Mule, Synapse and Celtix) – these products as well as components from the open source community covering UDDI, WSDL, WS-security etc. can in many cases fulfil the needs of an SMB⁶. It requires a careful analysis involving the resources available within the organisation, the external consultancy available, and the specific needs, to get an idea of the total cost of ownership. In particular since the ESB concept is by no means standardised.

While SMBs should embrace SOA as a pattern and many SMBs should have a strategy for introducing an ESB, developing a full service oriented client is not advisable for a typical SMB. The best current frameworks for developing an SOC requires substantial work as described in section 2.2.2. Furthermore, there are currently no standards on components for SOCs, and few (if any) business system vendors provide components which can be injected into Eclipse or NetBeans.

While an SOC embracing all the business functionality cannot be achieved by most SMBs, the less ambitious goal of using a portal framework for integrating key business processes and giving a cross-system overview of the business, is highly recommendable. A business process is suitable for integration into

⁶ JBoss ESB is also part of Red Hat JEMS; a complete SOA platform

a portal if it is (a) used by many users, and (b) does not require heavy user interaction. Similarly, dashboards, KPI overviews, business intelligence etc. are good candidates for portal components.

A portal approach will give the advantage of the SOC to many users in many situations. Users with heavy interaction with the system or users that occasionally have special need must use dedicated clients provided by the system vendors. While this is not an ideal situation, it should be remembered that the dedicated clients are usually developed with exactly this purpose in mind – thus the heavy users will get a user interface that is optimised for exactly their needs. As the platforms for developing rich clients improve, it is likely that integrated SOC covering a selection of work situations will become a realistic option, thus replacing the portals with more responsive and user friendly UIs.

SMBs should embrace BPEL as a key technology for process automation and web service orchestration. BPEL is backed by all major SOA vendors, and allows a complete decoupling of systems (in contrast to systems calling each other directly using web services).

This can be illustrated by the following simple example: Suppose that customers are registered in your ERP system as well as in your CRM system, so every time a new customer is created in the ERP system, the same customer should automatically be created in your CRM system. In a simple SOA, the ERP system will call a web service in the CRM system every time a customer is created. Suppose you add a new bank system where customers also need to be created – now the ERP system must be modified to call web services in this new system too (top half of Figure 2). If you instead created a simple BPEL process for creating customers, the process can be initiated whenever a customer is

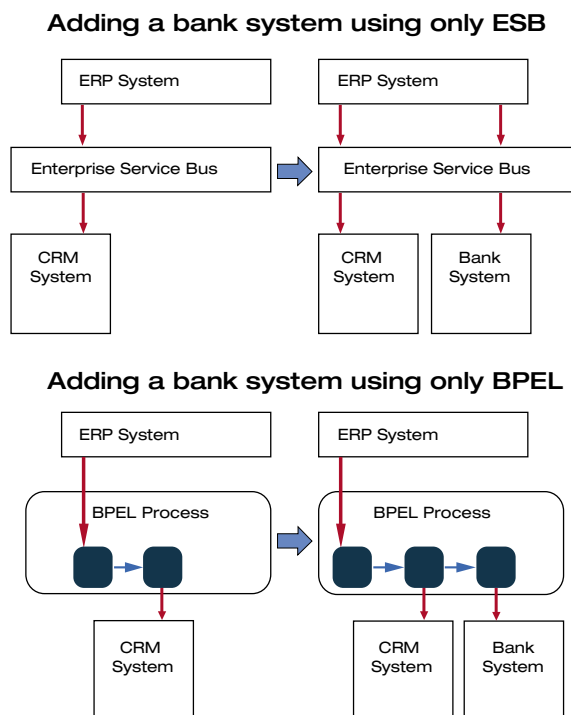


Figure 2

created and systems interested in new customers can attach themselves to the process (lower half of Figure 2). In particular, the ERP system does not need to be changed when the bank system is added. Such simple, automatic processes can be created with ease and need not involve complicated transaction control (see below).

This example illustrates how Web Services, ESB and BPEL are three steps of attaining a more and more loosely coupled IT infrastructure. Open source BPEL engines such as ActiveBPEL are available and provide a good solution to many SMBs.

There are a number of limitations to the use of BPEL:

- BPEL is a fairly low-level programmatic specification and requires programming skills. Many vendors provide more high-level Business Process Management (BPM) specifications that in principle allow non-technical people to modify business processes. The potential of this is huge, but BPM is not standardised and is still evolving quickly.
- BPEL provides strong functionality for automated workflows and processes, but is less suited for workflows that involve manual intervention. This is partly due to the design of the BPEL specification, but on a more pragmatic level, manual intervention in a BPEL process would either require the BPEL engine to have its own user interface or it would require a full SOC.
- When a business system delivers support for business processes, the user interface is often optimised to support the process. Furthermore, the business process will be an implementation of best practice and this best practice will automatically be maintained through upgrades at little cost to the user. Hence, processes that involve a single business system are best kept within the system. The process can, however, be utilised as a sub-process of a larger BPEL process.
- When defining business processes in BPEL, the system architect will need to consider transaction control – e.g. if one step of the process fails, what should happen to the steps already performed⁷. Transaction control (such as reverting an action) cannot be expected by all web services, so the process of defining the BPEL process might easily be difficult and technical.

To conclude, SMB should adopt coarse grained web services to detangle integration and to gain the benefits of standardisation. An ESB should be adopted to take detanglement further by centralising issues such as security and directory services. Finally, BPEL should be adopted for process integration to decouple systems completely – since each system only needs to be aware of the processes in which it takes part.

3.2 SOA Strategy

It should be clear now that the introduction of SOA in an SMB does not need to be a complicated and daunting task – instead it can be approached step by step. This way, the business will benefit from an SOA from the beginning while reducing the risk.

In this paper, we have not touched upon the organisational challenges carried by SOA – other authors have pointed out that to enjoy the full potential of an SOA, the strategy must be actively supported by the management and should not be a technical assignment for the IT department only – to benefit from the adaptability of SOA, business users should use the opportunity to modify and improve business processes as external conditions change.

Organisational issues will not be covered here, but it is important to emphasise that a clear strategy is crucial for the success of SOA. Such a strategy should clearly specify which parts of the SOA vision the

⁷ The archetypical example is a travel planner that will find the cheapest route between two points and then order the appropriate tickets from different airlines. If the system manages to buy one ticket but fails to buy the next, the first needs to be cancelled (typically this will not be the way to handle it – but the point is that the solution is technical and cannot be done independently of the functionality provided by the web service).

company believes will be beneficial for the company – and feature a roadmap for adoption. This enables the company to make an informed choice of infrastructure vendor (or open source) and qualify the need for consultancy. In short: start small, think big!

Having the fundamentals and the vision in place, the introduction of SOA can happen incrementally. This is true in two dimensions:

First we have described above how it is possible to move your Service Oriented Architecture from point-to-point integrations based on coarse-grained web-services onto more loosely coupled ESB as an architecture back-bone and/or use BPEL for process-oriented integrations. In parallel with this, a portal based user interface can be built to support the light-weight processes of the underlying SOA. Proceeding incrementally in this dimension does have a cost, and should only be adopted if the end-goal of the SOA strategy is not clear and an agile approach is preferred.

Secondly, SOA based integrations can be made one by one – this agility is one of the key benefits of the SOA platform, and should be taken advantage of even in the introduction phase. By doing the most important integrations first, big-bang investments can be avoided⁸. In addition, this approach facilitates learning, and integrations that initially require external consultants might later be handled in-house. This will make it easier for the IT department to support changing and improved business processes during normal operation.

4. Maconomy in an SOA

Maconomy believes in SOA for integration and has for some time offered functionality in MScript for exposing business functions as web services. As MScript also gives access to the full Maconomy dialog API and hence to the full business functionality of Maconomy, this provides an elegant and powerful way of publicising the Maconomy functionality as web services and makes Maconomy fit seamlessly into any SOA. Maconomy has deliberately chosen this approach over publicising the functionality directly as web services since such web services would be overly general to an extent where they are almost useless. With this approach, users can define web services that match the business processes of the users.

In the upcoming version X1 of Maconomy, this web service API is improved significantly:

- Native support of web service security (WS-security).
- Improved development environment, making it easier to use for programmers.

Maconomy sees SOA as a key driver for the architecture of future business systems. Maconomy is therefore devoted to continuously improving the functionality of the system in terms of working in an SOA environment. In particular Maconomy commits itself to:

⁸ A Fortune 100 company found that a "rip and replace" approach to introducing SOA would take 5 to 10 years and cost \$30-100 million simply preserving the present functionality. Instead an incremental approach was chosen, first replacing tight integrations with loose coupling based on services. The loose coupling facilitated additional steps where legacy systems could be refactored [5].



- Supporting the use of portals as light-weight SOCs by making key components from Maconomy's own portal available as components in the leading portal frameworks. At the same time, Maconomy is dedicated to delivering a state-of-art, dedicated Maconomy client efficiently supporting the work flows. This way, Maconomy users will get the optimal experience when using Maconomy, leading to the most efficient and effective use of the system.
- Providing first class consultancy on web service integration and SOA.
- Continuously improving the SOA Integration framework. Focus will be on extending tools and technology for data integration as well as business process integration based on SOA principles and standards. The SOA Integration framework will also be Maconomy's primary choice for integration within the Maconomy business system offering.

5. Conclusion

SOA offers great advantages for all businesses. The limited IT resources typically available in SMBs make it vital for the SMBs to use standardised integration methods based on proven practice – exactly what SOA promises. Nevertheless, SOA adoption rates are lower in SMBs than in larger enterprises. It is likely that this is due to the lack of clarity about the SOA concept and many products and standards, which makes SOA seem like a daunting (and expensive) endeavour for an SMB. This paper argues that by choosing an incremental approach to the introduction of SOA and by basing the SOA on a few widely adopted standards, the SMB can expect fast improvements from relatively modest investments. The core standards that the SMB should use are Web Services, UDDI, WS Security and BPEL. The technology for creating an SOC is still too immature, and for user interface, the best current technology is Portals, which are only suited for light weight interaction and business analysis.

By Christian Mossin, Maconomy

References

- [1] Randy Hefner: Survey Data Says: The Time For SOA Is Now; April 14, 2006, Forrester
- [2] Randy Hefner: Topic overview: Service-Oriented Architecture, May 26, 2006, Forrester
- [3] Steve Graham et al: Building Web Services with Java, 2005, Sams Publishing
- [4] Ken Vollmer and Mike Gilpin: The Forrester Wave™: Enterprise Service Bus, Q2 2006, June 30, 2006, Forrester
- [5] Bringing SOA Value Patterns to Life, June 2006, Oracle White Paper
- [6] Butler Group and Datamonitor numbers quoted on from <http://www.ebizq.net/>



www.maconomy.com

Copyright and Trademarks: © February 2007 Maconomy A/S. All rights reserved.
All product names and company names are trademarks or registered trademarks of the respective companies.