



# Service-oriented



# Architecture

Is Service-oriented Architecture (SOA) still a buzzword? Why would a company spend money to sponsor a so-called technical initiative by the IT team instead of financing some other business feature development by the team? Is it true that a lack of SOA in an enterprise with a complex IT ecosystem has a direct impact on the IT costs, and hence the bottom line, in the long run?

By Jayaprakash Nair, Delivery Manager, Aspire Systems

**L**ET US TAKE A quick look at a few things commonly observed in the industry today. **OBSERVATION 1:** Companies make investments in various “best of breed” products for their Line-of-Business (LOB) functionalities. These are typically monolithic systems tightly coupled together using traditional Enterprise Application Integration (EAI) approaches (and proprietary technologies to glue them together), leading to a heterogeneous and brittle IT ecosystem. Any seasoned industry observer will identify this as an omnipresent scenario.

**OBSERVATION 2:** The major software product companies had hitherto left gaps in their offerings, which were filled by smaller Independent Software Vendors (ISVs) with their niche offerings. (See *Diagram 1*.)

The following problems are likely to occur with this arrangement.

**Problem 1:** In due course of time, the company may want to get the ISV’s product extended to replace part of the functionality of P2 that belongs to another provider, and the functionality that needs to be changed could be anywhere amidst the existing workflow provided by P2. So, there would be a need to build more interfaces between the products to ensure that the flow is seamless. (See *Diagram 2*.)

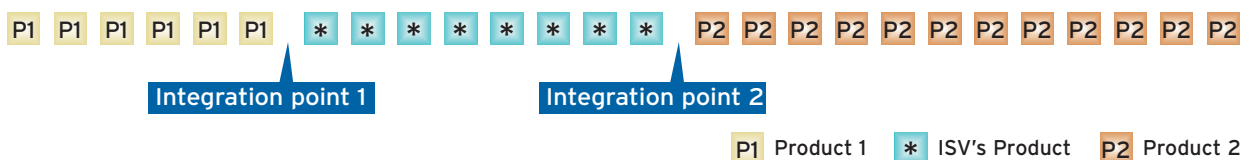
This leads to an increasingly fragmented IT ecosystem. You can consider this akin to the continuous fragmentation that happens on the hard disk because of deletion of some files/folders and addition of new files/folders which do not fit the size of an existing gap. This increased fragmentation leads to an increase in the cross-references between the dif-

Diagram 1

Gap in the eco-system:



Your product fills the gap:



ferent storage blocks on the hard disk, which is exactly what happens in a typical enterprise IT setup, leading to a spaghetti type of integration.

**Problem 2:** The company may decide to replace product P1 with another product (say P3), by another provider. There's a good likelihood that the company would need to approach the ISV to customize its product, or at least repair the integration point with the new product, in order to maintain the workflow/dataflow within the enterprise.

**OBSERVATION 3:** The company might introduce some changes in its business processes/rules, which would lead to further breakages in the integration.

**OBSERVATION 4:** In many cases, the trading partners of companies having business-to-business integration can change the interface points, thus breaking the integration.

As you would be well aware, the above observations are, by no means, hypothetical, and such occurrences are expected to increase, while moving forward. Let's see what can be done to alleviate some of the hurdles.

**The Solution — SOA**

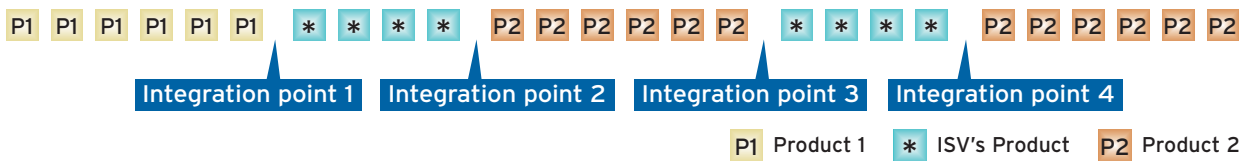
In the instances mentioned above, the existing IT ecosystem in an enterprise had potential breakage points. From a solution perspective, the firm will have two options:

components/products use the same language to communicate with each other, and it becomes easy to continuously realign or “re-integrate” them. This, in turn, leads to a very “amoebic” enterprise architecture, i.e. an architecture, which can continuously change shape, without any harm to its fundamental constitution.

**CUSTOMIZATION/ENHANCEMENT:** A typical software application solves one or more business problems, and business problems are manifested in business processes. An application can be effective only if it has a direct relevance to the business processes followed by the company, and most of them are likely to change very frequently. There are two problems identified in traditional product implementations:

**Problem 1:** Atomicity. The atomic level of a traditional application is normally a module, which is nothing but a set of closely knitted classes (here we assume that the application is developed in a disciplined object-oriented manner with the right composition of modules). Modules are typically significant in size, highly cohesive and put into production after different levels of rigorous testing. Now if there's a change in any of the business processes contained within a module, it would mean that the module needs to be broken open, possibly gutted and reworked on. It would require multiple analyze-code-test loops for the module to

Diagram 2



- Either continue with the existing IT setup and keep on increasing the ever-so-brittle integration interfaces between the different products/applications, thus increasing the complexity of the integration/implementation of the IT ecosystem.

- Or, convert the enterprise architecture into a SOA.

From the options, the first one is not really a “solution” and just amounts to maintaining the status quo (or maybe worsening the situation), whereas the second option is a solution that the company can adopt. Service-orientation describes an architecture that uses loosely coupled services to support the requirements of business processes and users.

**How SOA Can Reduce Costs**

**INTEGRATION:** A successfully implemented SOA will reduce the coupling between the different components in the system. This is possible because of the standards based (typically XML) interfaces that SOA adopts. This manner of using services for integrating the disparate systems is called Service Oriented Integration (SOI). This approach is totally different from the proprietary interfaces used by traditional EAI.

What happens in an SOA scenario is that the different

be put back in production. And then, there's always the huge risk of regression errors (and subsequent negative ripple effects) creeping into the module.

**Problem 2:** Non-segregation of variable and fixed business processes. Every business has a set of core business processes, which are relatively fixed in nature, i.e. they hardly change over a period of time. Then there are business processes that change very frequently. Now in a typical traditional application, modules are structured around features, and as such, contain a mix of business processes, which are fixed, and the ones that are not. Thus, every time a change is required in a business process, a module needs to be dismantled and reworked on.

Both the above problems can be solved, if the application adapts to an SOA. SOA requires that the code implementation of stable business processes be segregated from those, which change very frequently. Also, it requires that the atomic unit of cohesiveness be the “service” and not the module. This makes the whole application easily customizable, thus reducing customization/enhancement costs.

To sum it up, as the complexity of the IT ecosystem in a company increases, (primarily because of heterogeneity) the

revenue savings obtained from SOA also increases, provided SOA is implemented in a disciplined manner with proper governance in place. Also it has been observed that the IT ecosystems are indeed getting more and more complex.

Now let's take a look at the first steps a company can take to implement SOA.

### Call for Action

As a first step, you will need to talk to SOA consultants/providers and identify those with the required capabilities. Here are some of the things that you could keep in mind for narrowing down your list of prospective providers:

- Check if the provider has any standing in the IT-services space, since that will be crucial for your provider to understand your stated as well as unstated challenges, and work out solutions to alleviate those.

- Does the provider have consulting capabilities? Can they identify the cross-section within your enterprise where SOA can be implemented end-to-end with maximum impact? Note that at this stage, your provider should be focusing mainly on business process, and not too much on the technology.

- Check if that provider has a solid implementation team, because once the candidate cross-section for (converting to) SOA has been identified, and the architecture is in place, the next step would be to actually implement the solution using services. Here, you need to ensure that the provider has a proven track record in the requisite technologies.

- Does the provider have proven capabilities in the appropriate software-development methodologies (engineering as well as project management), which are apt for SOA implementation? For instance, using a waterfall model for this would be akin to fitting a round keg in a square hole. SOA and waterfall normally don't work well together. What you need is a provider who has sufficient expertise in Agile (or at least some other Iterative) methodologies.

- Any code that is developed needs to be tested. Now "SOA Testing" is a different beast. So check if your provider has that expertise.

- Check if your provider has a quality-management system in place to ensure consistently good quality.

- Needless to say, check if your provider will do all these at a low cost.

One option is to go for an amalgamation of different providers to meet the above goals, but you then need to be prepared to shell out the extra cost involved in project management, communication management and risk manage-

ment. Also, you need to be aware which provider you will hold responsible in case of any issues.

In short, you need to find a provider who has the ability to provide you with a "cost-effective" and "pragmatic turn-key" solution without disrupting any of your existing business activities.

### Different Approaches

There are three approaches that companies typically consider, for SOA adoption:

**TOP-DOWN APPROACH:** Here, the company builds an SOA framework suitable for their requirements (in terms of governance), creates services which cater to specific business processes, and then plugs these new services into the framework. The cons with this approach are the same as what has been seen with the waterfall model of software development.

- By the time the framework is built, some of the assumptions/policies of the company's business processes would have changed, and the framework would need to be reworked.

- It takes a lot of time to actually see any tangible result in the "SOAfication" of the enterprise.

**BOTTOM-UP APPROACH:** In this approach, the company starts at the leaf level, i.e. builds services, and then tries to stitch them together and arrive at a framework. With this approach, the risk is that, being focused on the trees, it is easy to lose sight of the forest. So, instead of an SOA, one could very well end up with Just a Bunch Of Web Services (JBOWS), without a uniform governance framework, which is so critical for any SOA implementation to be successful.

**"START IN THE MIDDLE AND SCALE OUT" APPROACH:** A complete overhauling of your architecture may not

be feasible. But will the evergreen Pareto Rule be applicable to your product? That is, will converting 20 percent of your product into SOA reduce 80 percent of your integration pains and, if yes, will the ROI justify the 20 percent investment?

Here, SOA is implemented in a cross-section of the enterprise, i.e. in a complete end-to-end LOB chain. A good cross-section of business-workflow is identified as a candidate, the required governance framework is built only for that section, and the corresponding services are designed, built, tested and deployed. This approach overcomes most of the cons of the above two approaches.

Normally, the last option is the best — start small, identify your main bottlenecks, hit at those places with well-governed services, see the results, and then extrapolate. **GS**

**AS THE COMPLEXITY  
OF THE IT ECOSYSTEM  
INCREASES, THE  
REVENUE SAVINGS  
OBTAINED FROM SOA  
ALSO INCREASES,  
PROVIDED SOA IS  
IMPLEMENTED IN A  
DISCIPLINED MANNER  
WITH PROPER  
GOVERNANCE IN PLACE**