SOA: Beyond the hype

Anis Benyelloul
anis.benyelloul at gmail.com

July 2008

Abstract

As the business ITs of today have become more and more demanding and challenging, Service Oriented Architecture (SOA) is emerging as the new paradigm that will enable software systems of tomorrow to rapidly evolve and adapt our ever changing needs. Along with this growing interest is a cloud of misunderstandings and preconceived ideas about what SOA really is, and how does it differ from what we've been doing before. This article will try to shed some light on what's really new about SOA, and how does SOA relates to the previous paradigms.

1 Introduction

Summary Lack of understanding of the fundamental concepts underlying SOA is one of the most important obstacle to its wide approval.

Nowadays, and more then ever before, business IT is put under high competitive pressure, and software systems are now required to adapt quickly and meet the requirements of an ever growing and fast changing market climate [2].

In this context, Service Oriented Architecture (SOA) is perceived by many as the software engineering paradigm for the next generation of business software. One that will permit rapid development, as well as fast re-configuration and re-composition of existing software assets.

On the other hand, it seems that the most important hurdle that could prevent wide adoption of SOA is actually a lack of understanding about the real concept behind it. The next sections will try to show why as well as give some insight to help establishing a better understanding of the most fundamental ideas.

2 Lack of focus

Summary Most introductory material put the focus on non-distinguishing properties of SOA that do not set it apart from older paradigms.

It seems that the main source of misunderstandings about SOA comes from the fact that most introductory resources on the subject put the focus on the wrong things, or do not insist enough on the actually relevant and distinguishing concepts that underlie SOA.

Many resources try to define SOA by defining what a service is, and they often end up with statements such as:

> A service is the fundamental, and primitive component of SOA. A service fulfills a function that is well-defined, self-contained, and does not depend on the context or state of other services.

This is correct, but the same definition could apply to the classical "procedure" of the procedural programming paradigms of the 70s, and thus it does not tell you much about what SOA is.

Other possible definitions of SOA include:

> SOA is about decomposing applications into loosely-coupled parts (called services) that

> have a clearly defined interface which promotes evolution, reusability and composability.

Again, this is true but not different from what Object Oriented Programming has been trying to achieve since its introduction in the 80s.

Other papers yet introduce Web services along with the definition of SOA. But as many authors have already noted [1]:

- There is nothing new about the concept of Web services, it is just a new form of distributed computing based on open standard (like CORBA).
- When it comes to SOA, Web Services are just one possible implementation platform, and certainly not part of the concept of SOA itself.

Some resources also try to include business process management concepts into SOA:

> SOA purpose is the control business processes, establishing corporate-wide security, privacy, and implementation policies, and providing auditable information trails, are all examples of ways that SOA can reduce several of the risks facing companies today.

The companies that actually didn't have any form of privacy, security and implementation policies, even long before the advent of SOA, were very seldom. Besides, business process modeling is just one possible application domain of SOA, and again not part of the concept of SOA itself.

All in all, among the most common mistake we make when trying to introduce SOA is to put the focus on:

Properties of Service Oriented Software such as loose-coupling, reusability, clear interface/implementation separation, abstraction... these are actually the well known properties of "good" software design that are carried over to the SOA world, but they should not be used as a foundation for understanding SOA.

Web Services Web services is just one possible *implementation platform* for SOA, and does not bring any new ground breaking ideas compared to existing distributed application middlewares[1].

Business Process Management Which is one possible application domain for SOA, and thus should not serve as a base for explaining what SOA is.

As a result of this confusion some authors arrived at the conclusion that there is nothing really new about the concept of SOA, that it is just a "hype" or a yet another technology buzzword[3].

3 SOA

Summary The objective of SOA is application reuse. The approach is to expose application functionality as services for other applications.

In order to precisely yet fully understand the concept, it useful to split the problem into two parts: the objective of SOA and the approach of SOA.

3.1 The Objective

The "objective" is what SOA strive to achieve. The visionary promise of SOA is a world were one could develop new applications by leveraging existing ones. This concept comes as the logical evolution of previous re-use efforts that were first trying to reuse procedures, then procedure libraries (e.g numerical analysis libraries), then classes and class libraries (e.g GUI toolkits), then came the component based software development (CBSD) paradigm which promotes the idea of being able to build software by *easily* assembling existing components without having to write much code yourself. The next step in this evolution is to be able to reuse entire *applications*.

Re-using applications does no mean that you "download" them (or part of them) and link them into your own (this would be traditional class library or CBSD reuse). It means that the applications are already deployed and used, that their maintenance

and evolution is under the responsibility of other people. One just has to leverage that application (where is it) and use its functionalities for his own purposes.

The fact that you can also reuse application across organization boundary, means that your own software would be composed of parts that are actively run and under control of third parties. This means that application development time can be dramatically reduced (you now have a much larger scope of existing software to choose from). But this obviously comes at the cost of a lot of issues (both technical and non-technical) that are beyond the scope of this article.

SOA is also different from traditional distributed computing. In the sens that, in the latter, you create an application that you "*split*" across a network of hosts. SOA adopts almost the opposite approach where you create new applications by "*assembling*" remotely available functionality.

3.2 The Approach

The "approach" defines *how* does SOA try to achieve its objective. This is where the notion of "service" come in. In order to be able to re-use existing applications they must *expose* their functionality as externally visible "services", which are simply a set of "*operations*" or "*functions*" (function names and arguments). This means that software developers must now integrate the idea that their applications could be remotely re-used by other ones, and thus make their design in a such a way that they can expose key functionalities as services.

4 Conclusion

Summary Having a good understanding of the fundamental concepts behind SOA is critical both for industry adoption and for future research development.

In order to make the definition more comprehensive we can add the following points:

- Web services are currently the most common and widely adopted implementation platform for SOA software systems. They define a common set of file and message format, that enables the inter-operability of applications running on different environments/platforms (e.g integrating a Java EE application with a .Net one)

- The most promising application domain for SOA seems to be business process modeling. Where each service becomes the entry point for the implementation of a business process.

- The fact of having to think of applications as composed and from different other applications that are run and maintained by third parties (possibly in an other organization) actually implies that the interfaces between the applications need to be designed in a way that permits the application evolution without breaking the clients. Which in turn means that it will most likely lead to clean and well designed architectures: these are the good "side effects" of SOA.

From what has been seen, it is necessary to distinguish between the different ideas revolving around the notion of SOA if we want to be able to pin point the essential concepts behind it, the ones that set it aside from the previous paradigms. This is essential to form a sound base of understanding that will enable both industry wide adoption and research development.

References

[1] Leymann, F. Web services: Distributed applications without limits. In *Datenbanksysteme in Büro, Technik und Wissenschaft - BTW2003* (February 2003), G. Weikum, H. Schöning, and E. Rahm, Eds., vol. 26 of *LNI*, GI, pp. 2–23.

[2] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F., and Krämer, B. J. 05462 service-oriented computing: A research roadmap. In *Service Oriented Computing (SOC)* (2006), F. Cubera, B. J. Krämer, and M. P. Papazoglou, Eds., no. 05462 in Dagstuhl Seminar Proceedings, Internationales

Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany. <http://drops.dagstuhl.de/opus/volltexte/2006/524> [date of citation: 2006-01-01].

[3] Vasters, C. Soa doesn't really exist, does it? http://vasters.com/clemensv/default,month,2005-05.aspx, 2005.