

# The SOA Magazine

## Feature Article



### Enterprise Mashups Part II: Why SOA Architects Should Care

by Chris Warner and John Crupi

Published: August 13, 2008 (SOA Magazine Issue XXI: August 2008)

[Download this article as a PDF document.](#)

*Abstract: Gartner recently named Enterprise Mashups a “Top 10 Strategic Technology for 2008”, noting that “by 2010, Web mashups will be the dominant model (80 percent) for the creation of composite enterprise applications.” [REF-1] This should make any SOA architect sit up and wonder: Can I describe the value of mashups? Can I outline the relationship between mashups and existing enterprise technology like SOA?*

*Knowing the answers to these questions can advance you well down the road to embracing this exciting technology in your organization. In [Part 1](#) of this three-part series, we defined a mashup in the context of the enterprise, contrasted it against other common data integration technologies, and outlined some of the more important architectural elements of an enterprise-grade mashup solution. Now, in Part 2, we’ll discuss why SOA architects should care about enterprise mashups.*

#### Introduction: The SOA-Mashup Conundrum

To understand the value and relationship between enterprise mashups and SOA, it is helpful to first understand why we need enterprise mashups at all. To recap what we covered in the preceding article:

- mashups give you faster answers
- mashups improve your resource use (of both personnel and soft/hard computing resources)
- mashups help you address new business opportunities by letting users assemble internal and external data in an opportunistic way

In a mashup world, SOA can provide the “service cloud” that supplies the raw materials to a community of mashup users. And therein lies the conundrum: The most successful SOA initiatives will drive the need for mashups, but the bottom-up, ad-hoc nature of mashups can run contrary to the very principles that made your SOA a success in the first place. Avoiding this trap is not impossible. But adding mashups to a service-oriented enterprise architecture does require that the SOA architect pay close attention to the “big 3” principle issues of SOA.

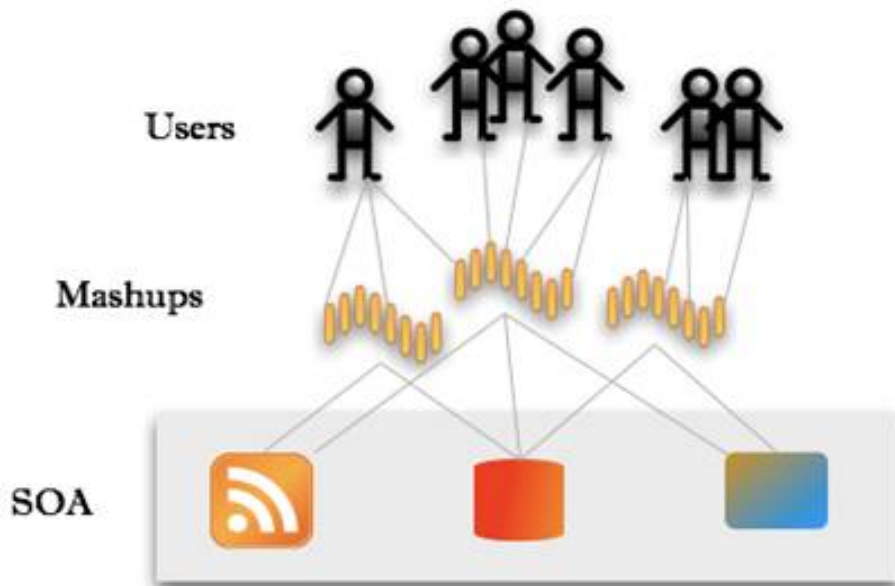
Having spent many years evangelizing, architecting and writing SOA solutions, we now know SOA is an easy sell to IT because it’s an architectural best practice. The service-oriented architectural style has attractive principles like shared services, loose-coupling and service reuse. We’ve also learned that IT is fast to accept SOA but can often get tripped up implementing the fundamental issues of governance, granularity, and scope.

The beauty of loosely-coupled services is in being able to decouple the service contract from the service implementation. Loose-coupling is like giving someone an address to your home and letting them get there anyway they want. But, what if you wanted them to only take a specific route and you wanted to check their ID before they drove up your driveway (doesn’t sound too inviting, but you’ll get the idea)? The same applies to SOA. Inherently, SOA does not force you down a specific route, nor does it check your credentials. Or in other words, there is no governance built into SOA. This is okay behind the firewall; but, when you want to expose services to the outside, IT gets very

nervous and all of a sudden loose-coupling looks a bit more dangerous and governance looks a lot more important. Thus, governance becomes paramount but it makes a simple path to SOA a bit more complicated.

After governance, service granularity peeks its head out. To get the SOA right-sized, you must address questions like “how large is a service”, “is a service atomic or does it represent a process”, “is a service so flexible it can support many variations or is it static and supports only one mega definition of data”. Granularity discussions can get heated with epic debates lasting years in some organizations. Ultimately, you’ll probably settle on “business granular”, meaning that you should gauge the service granularity by being able to talk about it in business terms. Think ‘Inventory’, ‘Payroll’, and ‘Orders’. If you can talk about both in the service and the data that is being exchanged in business vernacular, you’re on the right path. As you can imagine this is no small feat and, like governance, adds another layer of complexity to the SOA roadmap.

The final issue of scope determines how big or how much of a problem IT is willing is to tackle when SOA-tizing the datacenter. From a project management and funding perspective, scope can really impact the likelihood of success or failure. After years of brilliant marketing, CIOs are betting on the lofty ROI, agility and time-to-market promises by making SOA a top 3 corporate initiative. Good for funding and commitment; bad for pragmatic realization. We now know that SOA can be very successful if managed as a series of well-scoped, smallish projects that are tied to business need. In essence, proper scope can lead directly to SOA success...or failure.



*Figure 1: How mashups establish a layer between SOA and users.*

After navigating the governance, granularity and scope issues of your SOA, you have one (or more) small, successful SOA projects that have resulted in a library of business-driven, micro-services that have can immediate business impact. But there’s the rub in your successful SOA: You either have no direct demand for your services by business users or you have demand for direct access by business with no way of delivering them safely.

That’s where mashups come in. A mashup can help deliver a new, dynamic, and potentially huge group of consumers to your services. And that’s why an SOA architect should care about mashups: Mashups can become that invaluable direct link between an SOA and the business community of an organization.

But how can we push our services out to a mashup community without trampling the principle issues of governance, granularity and scope that made the SOA effort successful in the first place? And furthermore, what are the interaction patterns between your SOA and these mashups? It is this fusion of SOA and enterprise mashups that SOA architects must understand.

### **Mashups Can Deliver SOAs to the User (Without Breaking Your SOA)**

Mashups bring a “user” into the SOA mix. This is an entirely new element and one that is dynamic and unpredictable. But the SOA architect needs to appreciate this as a good thing: The new demand that is being generated for an SOA from a mashup community. By having the business build mashups upon the foundation of you service-oriented

architecture, they essentially become SOA champions without knowing it.

To get a better picture, it helps to consider some of the more common “connections and interactions” between the two:

### *Information Right-Sizing*

Three years ago we wrote a blog entitled: SOA Best Practice for Business Unit Alignment [REF-2]. The premise was simple: You had a better chance making your SOA successful if you included the business unit and made SOA more top-down. At the time, ~80% of the readers agreed and ~20% didn't.

The group that agreed not only believed this to be true, but felt this was the only way SOA would achieve success. They felt that without the business, IT would not be able to correctly define the correct service granularity. That also meant you should be able to talk about services in business (not technical) terms. Interestingly, the 20% that didn't agree felt that it should actually be done both top-down and bottom up.

But our point went further. IT needed to not only see the value in bringing the business to the table, but should actually let the business take more of a leadership role in defining the data they want and have that drives the service definition and creation. As a result, it is three years later and I have concluded that both groups were right.

The ideas are relevant here because mashups are a great forcing function for the right-sizing interaction between SOA architects and SOA business consumers. In most service-oriented solutions, IT needs to “right size” services. This kind of dynamic virtualization is best defined by the use-cases from the business but needs the guiding hand of IT to make it stick.

### *Publishing and Syndication*

Many view mashups as data displayed on maps. While the majority of early mashups certainly seemed to fit this description, today enterprise mashups can be published to many kinds of destinations that an enterprise user would appreciate. In addition to SOA-friendly formats, such as RSS and Atom plus REST and SOAP, mashup creators can publish mashups to spreadsheets, as WSRP-compliant portlets, wiki- and blog-friendly widgets, or even into a mobile phone as a micro-application. Mashups can become the vehicle through which services become part of the everyday tools of the enterprise business user.

A detailed example can reinforce this idea. Certainly most Web surfers have seen the power of syndicated widgets in popular sites such as iGoogle, Netvibes, Yahoo Maps and YouTube. Widgets let users do one or two narrowly-defined things well. And because both widgets and mashups are micro in scale, the fusion of widgets and mashups can yield explosive results. Today, the newly-combined data in your mashup can be published as a widget for use by others.

One good example of mashup-driven widgets is the system called “The Badge” from Thomson Scientific's ResearcherID.com community [REF-3]. The Badges let research professionals around the world publish a dynamic set of personal research data (such as number of citations, citations by country, and other professional data) into their own personal blog or Web site. To the Web site visitor, the Badge looks like a graphical button; it is in fact a widget that communicates with a service-oriented solution in the Thomson datacenter.

### *Mashup-to-SOA Publishing*

The whole value proposition of loosely-coupled services gets muddled by the need to version services. The public interface of services, once published to the greater consuming community, must remain static. You're pretty much stuck with them. Yes, you can introduce bug fixes, but fundamental enhancements that affect the public service definition (like changing data types and adding new operations) can be very problematic for services that are widely consumed. Popularity, in a sense, can be an impediment to improving your SOA.

This is where mashups can directly help the SOA architect and developer. Mashups can go well beyond leveraging an SOA by becoming part of that SOA, allowing developers to create customized “service skins” from core services. These skins provide a thin, easy-to-manage buffer against frequent or dramatic changes to the core services within your SOA.

Because mashups can be exposed as REST-, WSDL- and JSON-based services, they look and feel like a real SOA-based service to developers who want to consume them. In this use case, the service(s) from which the mashup is

derived remain unchanged and become more of a “behind the scenes” core service. The mashup, created by the developer, becomes the tailored service which is directly aligned with their particular need. Major enhancements to core services can be accommodated with a reformulated or updated mashup by the mashup creators themselves.

### *Service Virtualization*

At a recent conference we asked the audience how many have put SOA into production. Many waggled their hand too-and-fro, not indicating a definitive yes or no but a “sort of”. As most SOA architects know by now, implementing a service-oriented solution can become quite complex. That means good SOA doesn’t happen overnight and most SOA teams are diligently working their way through a very long list of services that need to be constructed. Mashups can help create quick-and-dirty “virtual” services from sources that haven’t yet been service-oriented. Until the formal SOA magic has been applied across your enterprise, a good mashup tool can provide a light-weight, normalized virtual service for mashup users.

More inquiry with my conference audience revealed that most were not completely comfortable saying their SOA was “in production” for services that just did one or two things. They were under the impression that SOA meant “big in size” and “big in functionality”. Or, said another way, functionally-limited services didn’t jive with their view of a properly-architected SOA. But many admitted that there were many services that were essentially simple and data-centric and were being highly utilized by the business.

This is a great opportunity for SOA architects and developers to add mashups to their bag of tricks. Since mashups provide data services that work best by exposing discreet sets of data, it makes sense to employ mashups as a data service provider within your SOA when the service is small and specialized. Sure, you could still have the big services underneath, but mashups make a great light-weight solution that can reduce the services’ complexity and help expose right-sized virtual services in manageable, easily-consumable parts.

### *Inside-Outside Combinations*

The trend towards standardized data formats makes it possible for mashups to combine your SOA with public or external sources that have adopted a standard data exchange format. Imagine you have customer data in your internal Siebel database and more customer data in a hosted Salesforce system. To get a list of opportunities from both systems and view it as a single data source creates a data migration problem. Instead of selecting one of the systems as the master data provider and moving the other data into it, a simple mashup that takes advantage of the standardized service interface of both systems allows you to create a mashup from them. (This assumes you’ve added that nice set of SOA-based services to your internal Siebel system, of course!). In other words, mashups let you create dynamic formats aligned with the users needs without having to migrate large chunks of data.

### **Conclusion**

A mashup can be a first-class service consumer. But a recent Forrester report emphasized that a proper approach to enterprise mashups must “provide a safety net for business users” [REF-4]. In other words, your mashup efforts must address governance, granularity and scope issues as proactively as your SOA efforts did. Assuming you agree and desire this new consumer base for your SOA, the natural inclination might be to rework your original SOA governance, granularity and scope to make it more “user” centric.

At this point any SOA architect worth his WSDL should see that mashups can greatly enhance an SOA and don’t necessarily ignore or break the principles that made your SOA great. But governance, granularity and scope for mashups do require subtle tweaks and adjustments to your enterprise toolset. Because the business has different needs. They want small, not big. They want micro-slices of data, not all of it. They want combined data from a small number of services so that they can further combine it with external systems.

Fear not. Mashup governance, granularity and scope is not as difficult as it might sound and it need not be as difficult as it was during your SOA. If properly architected, your enterprise mashup solution can leverage many SOA technologies, acting as a peer to your SOA. In Part 3 of this series we’ll do just that: discuss an enterprise architecture that incorporates mashups as part of SOA-enabled ERP/CRM/SFA/BI and homegrown applications.

### **References**

- [REF-1] [Gartner Identifies the Top 10 Strategic Technologies for 2008](#), Gartner
- [REF-2] [SOA Best Practice for Business Unit Alignment](#) by John Crupi, JackBe
- [REF-3] [Mashups in Action: Fusing Enterprise Mashups to Enterprise Widgets \(The Rest of the Story\)](#)  
by John Crupi, JackBe
- [REF-4] [Enterprise Mashups: Lead, Don't Follow](#) by Mike Gualtieri, Forrester

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2008  
SOA Systems Inc.