

The SOA Magazine

Feature Article



Enterprise Mashups Part I: Bringing SOA to the People

by John Crupi and Chris Warner

Published: May 16, 2008 (SOA Magazine Issue XVIII: May 2008)

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

Abstract: Forrester Research predicts that mashups will be a \$682 million industry in the next 5 years [REF-1]. But can you define mashups? Can you describe the value of mashups to an SOA architect or even a business user? Can you outline the relationship between mashups and existing enterprise technology? Knowing the answers to these questions will advance you well down the road to embracing the concepts and techniques behind mashups in your organization.

This three-part series will help you get a head start by discussing the gritty details. In Part 1 we'll define a mashup in the context of the enterprise, contrast it against other common data integration technologies, and outline some of the more important architectural elements. In Part 2 we'll discuss why SOA architects should care about enterprise mashups. Finally, in Part 3 we'll discuss an enterprise architecture that incorporates mashups as part of your SOA-enabled ERP/CRM/SFA/BI and homegrown applications.

Introduction: What's a Mashup?

Everyone seems to have a gut "feel" for the term, but many can't seem to put their finger on it. According to an Economist Intelligence Unit survey from January 2007, 64% of companies are already adopting mashups or plan to within the next two years. Much of the education and early success has been with consumer mashups, such as the Chicago Apartment Locator, better known as www.housingmaps.com (and www.zillow.com).

Wikipedia describes a mashup as "a Web site or application that combines content from more than one source into an integrated experience." But defining an *enterprise* mashup is another story. While it's a good start, this definition doesn't address all the 'secondary' requirements inherent in an enterprise, including security, governance, compliance, and integration with other tools. Connecting your SAP ERP data with your Oracle/Siebel CRM data and third-party demographic information, all while maintaining single sign-on usage into your global LDAP server is not the same kind of problem as showing a map of apartments for rent in Chicago.

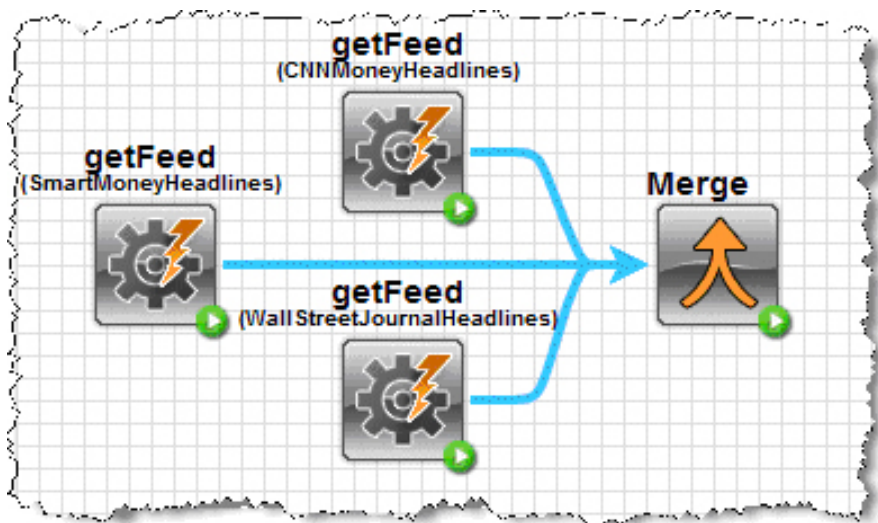


Figure 1: An example of how a mashup is comprised of different sources.

A definition of a mashup that matches the complexity of the typical enterprise would be "a user-centric micro-combination of standards-based internal and external data sources". This definition contains a number of important points worth discussing in greater detail:

User-Centric

Mashups are always for the end-user and often created by the user themselves. (IT is still involved in mashup creation but we'll get to their role a bit later.) The recipient of a mashup is also another user like a co-worker, a partner, or a customer. Finally, the mashup data sources are generally user consumable. This means the data makes business sense and doesn't require large IT systems to clean and normalize the data.

Micro-Combinations

Think of a user copying data from Web sites or other applications and pasting it into Excel. These users are doing the equivalent of a low-tech mashup that serves as a great model for the modern approach. These users typically deal with small amounts of knowledge-oriented information (as opposed to IT-managed applications that typically deal with large amounts of transactional information), which is why we call these "micro-combinations."

If we had to put a number on it, it's more than two data sources and most likely around four to six data sources being combined. Also, the amount of data being mashed-up is also small. It's in the tens, hundreds or thousands of rows, not in the hundreds of thousands or millions range. (Those larger volumes are what ESB and ETL engines are for.)

Standards-based Internal and External Data

Mashups are predicated on the wave of standardized data and access formats that are washing over us including WSDL, REST and RSS. We summarize these here as "Web-accessible." In other words, our data sources shouldn't require too much manipulation for the user to make sense of it.

So, now that we've formally defined it, is this definition enough for us to truly understand an enterprise mashup? Well, it's pretty much comparable to describing a car as a 'passenger vehicle designed for operation on ordinary roads and typically having four wheels and a gasoline or diesel internal-combustion engine' [REF-2]. We are left to wonder if our 8th-grade go-cart fits the bill. To give you more insight into real world mashups, we've captured some of the common enterprise-related characteristics that help expand on our definition:

- *Collaborative* - Mashups are designed to be tagged/searchable/shared with others. User tagging, often called a 'folksonomy', helps users put meaning for themselves and others.
- *Focused on the 'pack'* - Mashups are typically created, used, and shared among a small number of related individuals. Knowledge workers collaborate in small packs. Although they may be part of a larger group, they usually function as small teams when it comes to discrete information needs.

- *Time-sensitive* - Users need data now. Mashups usually have near real-time delivery requirements. They don't have time to wait for IT to "pre-integrate" data so they can get at it. The Web is real-time and business users have evolved to expect the same inside their enterprises.
- *Non-invasive* - There's no need to bring in a whole new set of infrastructure, as enterprise mashups run inside the current enterprise stack. This includes both mashup sources (databases, SOA services, etc.) and mashup destinations (portals, blogs, wikis, email, spreadsheets, etc.).
- *Limited cleansing* - The amount of data cleansing and normalization needed should be comparable to the amount of cleansing and normalization a user does in Excel. If there's more, you have a bigger problem that should be addressed concurrently with your mashup initiative.
- *Have a face* - Mashups usually have a face and the face is a widget. Just like mashups are "micro", so are the applications that front-end them. If the user is the recipient of the mashup, it's only natural for the user to be given a way to interact with the data.

But I Can Do That With My...

Our definition can also help us compare and contrast enterprise mashups with other common data integration tools. At first glance there are apparent overlaps with popular enterprise technologies like Business Process Management (BPM), Business Intelligence (BI), Enterprise Information Integration (EII), and even the SOA Architect's favorite tool, the Enterprise Service Bus (ESB). But a mashup, when applied appropriately, is none of these things and complements all of them. Let's outline some of the most important similarities and differences.

- *BPM* - Mashups aren't trying to address long-term workflow issues that involve human interaction. They are executed in seconds and don't contain human-interface breakpoints in them. This also means a mashup can make a great participant in a BPM process where automated data-driven decision points are called for.
- *BI* - We don't need a data warehouse to begin our mashing. Mashups impose a 'low footprint' and hence often connect directly to source systems that are also known to feed a BI warehouse/mart (which can, of course, also serve as a great source for a mashup). Furthermore, a mashup can provide high-quality input to BI front-end visualization tools.
- *EII* - Mashups aren't trying to address tough semantic issues that require sophisticated ontologies. Mashups are to be built, used (and possibly discarded) in near-real time. Much like BI, a completed EII service cloud serve as a great source for a mashup.
- *ESB* - Mashups are commonly constructed by end-users and are almost always delivered to other users (not other applications). An ESB can help expose archaic or complicated data sources as mashup-consumable sources. And, like BO, a mashup can provide great input for ESB tools.

All of these technologies have established value in the enterprise. Mashups support and extend them by introducing the ability to create dynamic, user-centric solutions.

What's a Mashup Good For?

It's interesting that while our definition (and most others) describes what an enterprise mashup is, it doesn't tell us much about its potential usage. While they are applicable to most industries and enterprises, trying to prescribe what mashups might be used for is like prescribing what can be built with the lumber. Your list would either be too limiting or it would be incomplete. The important part is realizing that the eventual use of the lumber (or the mashup) is up to you.

Let's now explore a real-world example. One mature and well-documented project is 'Overwatch', a real-time intelligence gathering and sharing interface built by the Defense Intelligence Agency (DIA) [REF-3]. The DIA moved from a cut-and-paste approach to a dynamic browser-based interface that allows intelligence analysts to connect live data sources together. Every resulting 'briefing' is based upon real-time, live information that can also be shared collaboratively among the analysts. Overwatch takes advantage of a sophisticated SOA infrastructure within the strict security confines of the DIA environment.

Here are a few more use cases:

- A portfolio manager making decisions based on mashing internal analysis data combined with Web-based financial and news data,
- Medical researchers extending their collaborative efforts by comparing and joining public research compendiums (such as Pubmed, www.pubmed.org) with internal research databases.
- A sales executive enhancing quarterly forecasting by joining local spreadsheet-based estimates with corporate application services like receivables and support history.
- A finance manager using aggregated, filtered, and combined data from across divisional accounting instances for intermediate or mid-term compliance and financial reporting summaries.
- A product manager connecting near-term production plans with real-time commodity prices to enhance production and purchasing decisions.
- A project manager establishing a sharable 'management view' of project deliverables, open issues, and personnel billing.

These examples all meet the benchmark of being 'user-centric micro-combinations of standards-based internal and external data sources'. Most importantly, each directly involves the user. Although IT doesn't prescribe the integration, it does play a very important role in the process, as explained next.

The Architecture Behind Enterprise Mashups (and the Role of IT)

If a mashup is all about the user, what does IT have to do with it? Like their Web 2.0 cousins, blogs and wikis, enterprise mashups require IT's guidance to give them structure and safety. That's fancy-sounding philosophy to be sure but what does this mean practically? The answer lies in the architecture of enterprise mashups.

Mashups can be created within the browser or on a server. Surprisingly, many of today's most popular mashups are browser-based, meaning that they do all the heavy mashup lifting right inside your browser. They query the data sources, perform the filtering and combining, and render the data in a pretty format...all with the confines of the browser. But these kinds of mashups assume a few things you won't find in any enterprise. Most make use of homogeneous, often security-free data sources that have relatively-recent data. In other words, the data is relatively clean and uncomplicated.

Furthermore, they assume that you, as the ultimate consumer, don't want to add, change or delete anything from the mashup. In short, this is nothing at all like real-world enterprise architectures. As one CIO put it, 'once your go down the enterprise rabbit hole, the rules are very different'. He couldn't be more right.

There are a number of issues that differentiate enterprise and non-enterprise mashups, most notably the disparate nature of the sources and the need for enterprise-grade security/governance. Applying these requirements to even a simple mashup leads most architects to a fundamental architectural conclusion: a server-based mashup solution is needed. Why? A simple example can help demonstrate this critical architectural decision point:

Consider a basic two-source mashup. Say you want to display on a map all Q3 orders from data currently stored in a Salesforce.com database. To make it more useful to you and your superiors, you also want to retrieve predicted Q3 inventory that is proximate to every order destination. For the sake of this example, our inventory is stored in an inside-the-firewall Siebel database. To do this, you (or more accurately, a developer) must write some JavaScript code to retrieve the orders from the Salesforce.com API, the inventory counts from the Seibel API, join them, use a third-party geo-coding service to add latitude/longitude, and pass it all to the Google Maps API.

Any JavaScript developer could probably work you up a prototype of this mashup in a few days. Forgetting for a moment that we didn't want to involve a developer in the first place (what's dynamic about that?), let's take a few more considerations into account:

- How do we (or our developer) navigate the security of Salesforce.com and Seibel?
- How do we know how to select 'proximate' inventory for each order destination?
- How many records are we going to be manipulating inside our browser's memory?

- Are we confident our corporate firewall is going to let some script from a user's browser make a call to an external source like Salesforce.com?
- How does IT monitor our mashup to ensure it's there when we need it?
- What if we need to add another data source?
- What if that source's interface isn't as simple as WSDL or RSS?
- What if we need to filter items before they are displayed?
- What if we need a table or a graph instead of a map?

To accommodate these concerns we need something that is user-centric and avoids the scalability, management, and security problems of a browser-driven approach. This is when a mashup server enters the picture.

Consider the following:

- The mashup server can have pre-negotiated contracts with the messy list of common enterprise data source types (SQL/DAO, WSDL, REST, RSS, POJO, etc.). It can even bring a little order to this cloud through a single 'virtual service' veneer, letting non-technical users mash sources without the need to understand the difference between acronyms like WSDL, RSS, and REST.
- The mashup server can provide a visual drag-and-drop interface for mashing our virtual services together. Common functions like inner-join, outer-join, merge, filter and even custom functions such as 'proximate' or 'geo-code' can be built into our visual mashup toolbox, again making it possible for laypeople to incorporate these fancy options without programming.
- The mashup server can safely store authentication and authorization information for virtualized services. It can also be connected to the enterprise's own authentication and monitoring toolsets.
- The mashup server can also help users share mashups through pre-built, server-managed interfaces via tools users work with every day: portal, spreadsheet, email, and the other data integration solutions we discussed earlier.

Are server-driven mashups required for the enterprise? No. Some trivial mashing could be successfully completed in a browser-driven architecture, albeit not by the typical member of the end-user community. But even a simple solution above becomes almost impossible to achieve once you've factored in the security and reliability requirements common to an enterprise.

IT's role in an enterprise mashup initiative is easy to understand. IT becomes the 'mashup provisioner', establishing the mashup server, feeding it formal services (and virtual services if needed), and connecting the mashup server to security, governance and monitoring tools. The road to successful enterprise mashup solutions is the same as any other enterprise solution; a proper, enterprise-capable architecture is a must and IT makes that happen.

Conclusion: The Bottom Line of Enterprise Mashups

The specific benefits of enterprise mashups will vary according to their usage. But some generalized benefits can be outlined. In a recent note Gartner summarized the benefits of mashups as 'speed', 'scale', and 'scope' [REF-4].

1. *Faster answers* - Mashups give the user-in-need access to both internal and external information without a middleman. Because of this, mashups provide answers faster than almost any other information management technique. For business scenarios, where performance is crucial, mashups provide the fastest way to access critical business information.
2. *Improved resource use* - Mashups give business users the ability to assemble custom situational information solutions, and they do so from data sources they may have previously used. IT is freed to address other tasks, while previously underutilized data sources become more relevant.
3. *New opportunities* - Because mashups let users combine data in new ways, they can support processes or decisions that were untenable even in the recent past. (If you haven't gotten your first "Long Tail" information request, now's the time to understand this type of information consumer.)

Gartner's Anthony Bradley summarized it nicely when he stated: "Application leaders and architects must investigate this growing space because of the enormous potential it may offer enterprises." [REF-5] We'll do just that in Part 2 of this article series, where we'll explore how enterprise mashups relate to and build upon SOA.

References

[REF-1] Oliver Young, Forrester Research, April 18, 2008

[REF-2] Dictionary.com

[REF-3] "Top secret: DIA embraces Web 2.0", Computerworld, February 2007

[REF-4] "Mashups and Their Relevance to the Enterprise", Gartner, September 2007

[REF-5] "Key Issues for Enterprise Mashup Practices, Technologies and Products", Gartner March 2008

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.