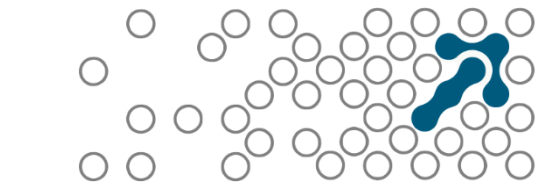
SHEDDING SOME MORE LIGHT ON THE "SHARED SERVICES" CONVERSATION

By: Dave McComb



semantic arts

Although there are at least seven levels of granularity to "shared services," little time has been spent to categorize them.

My observation is that although there are at least seven levels of granularity to "shared services," little time has been spent to categorize these. Please refer to the illustration below. The degree of sharing runs a gamut from the most sharing most at the top to the least at the bottom. Mostly the higher levels of sharing imply the levels below, but that's only most of the time, not all the time.

The colors could come in handy later to help visualize sharing by function and by agency in a large matrix.

An example might help--let's say we were trying to sort out shared services in the area of the motor pool. Let's go through each level:

The motor pool example doesn't quite do justice to the distinction between the application front end and the application back end, which we think may end up being the significant difference.

A larger and more traditional application may showcase that difference better. Let's take payroll. When most people talk about HR as a shared service they are talking about sharing the application (there hasn't been much discussion about the possibility of rebadging HR employees or relocating them).

So assume we're just talking about the HR application, there is still an extra degree of sharing to discuss; front end or back end.

Traditionally when you implement a package, like SAP, most everyone affected has to learn the new application. It has new screens, new terminology, new work flow, new exceptions and new conventions. It requires new interfaces to existing systems in the field. This is why packaged implementations cost so much. The software isn't very expensive. The literal installation and configuration doesn't take all that much effort. It is the number and degree to which people, processes and other systems are impacted that runs the price tags up.

For most of the agencies we have been involved with, HRMS was a wrenching conversion. Many have still not recovered to their previous level of productivity. But at least one agency that we know of had a pretty easy go of it. This is because they had built an app they called HR Café. HR Café was the interface that everyone in the agency knew and used. HR Café implemented many of their local idiosyncrasies. Almost no one had direct access to the old Payroll system. So when HRMS came up, the agency just changed the interface from HR Café so that it now interacted with HRMS, and there was very little collateral damage. The back end of HRMS was shared and not the front end.

In this case, the good result was sort of an inadvertent result of some other good decisions that were taken. But we think this approach can be generalized with a tremendous amount of economic benefit.

Headless Apps

We call this idea of sharing the back end, without necessarily sharing the front end, of an application "headless apps." It is a logical extension of good SOA design.

In good SOA design, most, if not all, the functionality of an application or shared service can be accomplished through the message interface. These (XML) message interfaces are primarily designed for use by other applications and services, but with a very small adjustment they can be the basis for composite user interfaces.

By concentrating on these message interfaces, coupled with modern browser based approaches to UI, it is pretty easy to both build a simple UI that maps straight to the message interface, or allow any agency that has a more complex existing UI to incorporate the new system in as a shared service. This works whether the shared app or shared service is a package or custom coded.

11 Old Town Square
Suite 200
Fort Collins, CO 80524

970-490-2224
305-425-2224
info@semanticarts.com