SupplyChainDigest™
The Supply Chain Technologist – April 14, 2005

Services Oriented Architectures and Supply Chain Software (Part 1)

By Mark Fralick
SupplyChainDigest Contributing Editor

If you've followed IT developments at all over recent years, you've will have seen much written on the topic of Services Oriented Architectures, or SOA.

A lot of experts now see the idea of a Services Oriented Architecture as the Holy Grail of software, speeding application development and (and, in combination with a specific implementation of SOA called web services) greatly reducing the time and cost of system integration. I agree that SOA is a game-changing concept, but for some additional reasons. SOA can enable a fundamentally different way of thinking about and implementing systems, and will be the foundation of a real-time, event-driven chain.

Think about an SOA this way: you essentially have a set of capabilities – all nicely encapsulated (exposed) as individual bits of functionality. They exist without the context of a specific application. In other words, an SOA revolves around the idea that functionality is exposed as a set of services that are not, in fact, directly connected to an application. These pieces of functionality, or "services," can be at many levels of capability or granularity, from performing a credit check or getting transportation rate, to printing a bar code label or encoding an RFID tag,

In some respects, there are parallels between SOA and the idea behind database management software, though SAO is at a much higher level. Before Data Base Management Software (DBMS), data storage and retrieval was tightly coupled to a particular set of applications. But the concept of DBMS brought with it strong separation between the Database and applications wanting to operate on that data. Think of SOA as the next logical step in evolution. An SOA provides you with much higher-level access to functionality.

Here is an example in warehouse management to demonstrate the differences: A DBMS provided us with an ability to say to the system "If I want to move this pallet from point A to point B, I'll need to ask the DBMS to update these 3 tables and perhaps insert data into this table." An SOA permits you to say something like this to the system: "If I want to move this pallet from point A to point B, I'll simply use this service". What makes this concept exciting is the fact that in order for you to work at the DBMS level, you had to understand the database layout (called a schema), invoke routines, call procedures or activate objects, and many other complex technical details. Those things all require understanding the specific context of an application. The SOA model, instead, says "Here are all the services you can use to operate your system." You need not be concerned with how they are implemented and what they are doing – you just know this is the service to use to move a pallet, perform a credit check, or calculate a cost. Behind the scenes the services may be doing a lot of low-level technical work, but the requestor of the service doesn't know or care what they are doing.

In an SOA, therefore, all the real ROI of the system is exposed as services to be used as many applications see fit. As an example of how this can benefit you consider that you may be performing all

receiving functions via GUI terminals today; but tomorrow you may want to receive onto a sorter. An SOA would give you the ability to do this using the same services, but invoked by the automation system. Yes, a vendor offering a product based on SOA will have the services configured to be called by client applications (called consumers, in SOA language), but the key is that the application side of the fence (the consumers) and the services are very separate. Indeed, if you wish some other piece of software – say your ERP software – to activate certain services directly, this should be quite simple to do (say, for example, to reserve inventory). This is a good example of how an SOA extends the ROI of your software and adds to its usable life.

The Next Step

All of that is nice, but not really the exciting part of what is possible once you have an SOA. As I indicated above, the real power comes from what you can do once you have a true SOA. Once services are separated from the application (they don't need an application to exist or run within), you have very powerful tools. You can, for example, start to consider the idea of executing services based on events. Again, we'll use a warehouse example. Consider a warehouse manager as he or she walks the facility. The manager sees some pallets stacking up in an outbound P&D (Pickup and Deposit) location. The manager gets on the radio and expedites some workers down to clear up the P&D. The event of the warehouse manager seeing the pallets stacking up triggered him or her to call over pallet jacks. An event-driven SOA gives you the ability to think of your system problems the same way.

Here are some examples of triggering services from events within a supply chain:

- When notified of schedule shipment being late (the event), automatically check for the impact of the late inventory on inventory deployment plans.
- When an inbound truck is checked-in (the event), activate the service that dispatches an RF equipped fork truck to unload it.
- When an outbound trailer is available (the event), activate the service that releases the shipments designated for that particular carrier.
- When an RFID tag is read in an outbound dock door (the event), activate the service that performs the movement of the material associated with the RFID tag to the trailer at that location. In this example, one could expect that some type of context, like an RFID vehicle tag, exists to provide feedback (good or bad) to the operator on the mobile terminal moving the product.
- When an international order is received (the event), automatically call the service which performance compliance checking for that customer/product/country.

Essentially, any event you can perceive becomes an event to which you can tie functionality. The power of this type of system lies in its endless ability to be individualized. Additionally, on the other side of the ledger - you shouldn't see the usual (and dramatic) costs for customization, risks of modification, and time delay if you are working with this approach. The key lies in the fact that all services have a well-known interface, a contract to provide certain functionality, and are fully tested as stand-alone components. Therefore, the act of individualization is really an exercise of configuration and system testing. An event-based SOA provides its user with the best of all worlds by:

- Providing for success today and tomorrow by providing a rich set of services that can be configured to fire from any event happening in the supply chain system system.
- Providing investment protection. Since the services are simply "black boxes" consumed by the requestor (applications for example), you care less about changes in technology as long as you can invoke them. The language they are written in, for example, or the platform they run on - could change from release to release. But since an SOA is a black box system – the requesting client application would never know this.
- Delivering systems that can be tailored to meet new business requirements with much less expense and risk.

- Extending the ROI and usable life of the software by creating an eco-system from which the true value of the services may be exploited. Services become useable by systems and resources other than just the application (business process management software, other enterprise systems, etc.)

The event-driven possibilities of SOA will really be the foundation of the "real-time enterprise," and a real-time, flexible supply chain that responds to customer demand and disruptions. Next time, we'll explore in more depth the relationship between SOA and "web services," and the promise of both to improve business processes and ease internal and external system integration.

See us at www.GetUsROI.com or email me at Mark.Fralick@GetUsROI.com

About the Author

Mark Fralick is president of ROI Solutions and a SuppyChainDigest Contributing Editor. Having operational, implementation and software development experience, along with detailed understanding the workings of software and services companies, puts ROI Solutions in a unique position to stand strongly on the customer's side in the battle for ROI in system selection and implementation. Learn more at www.GetUsROI.com, or email Mark Fralick at Mark.Fralick@GetUsROI.com.

About SupplyChainDigest

SupplyChainDigest™ is the industry's premier interactive knowledge source, providing timely, relevant, in-context information. Reaching tens of thousands of supply chain and logistics decision-makers each week, our flagship publications - *SupplyChainDigest* and *SupplyChainDigest – Logistics Edition*, and web site (www.scdigest.com) deliver news, opinions and information to help end users improve supply chain processes and find technology solutions.

For more information, contact SupplyChainDigest at:

937-885-3253
www.scdigest.com
email: info@scdigest.com