BPM and SOA

Mike Rosen
Chief Scientist
Wilton Consulting Group.

Mike.rosen@wiltonconsultinggroup.com

Orchestration or Choreography?

Now and then, we see discussions or debates about orchestration versus choreography. Often, it's no more than the standard 'my technique is better than your's', talking but not listening approach. Or, it's focused on a simplified differentiation between BPEL and WS-CDL aimed at making sense of why we have so many different standards (good question). Here, the typical line is that BPEL is for orchestrating a single process or composite service, and WS-CDL is for coordinating interactions between processes. And while that's essentially true, it doesn't get to the essence of the distinction. But recently, I saw a really insightful presentation on the matter by John Tibbetts, Senior Cutter Consultant, at the Cutter Consortium Enterprise Architecture Forum in Mexico City. This article summarizes the 'orchestration versus choreography' points of John's presentation "Using BPM and SOA to Improve Market Flexibility."

Let's start with several premises. First, services are a good approach to implementing IT functionality. Second, SOA is an architectural style, based on services as the fundamental unit for constructing enterprise solutions from services. Third, BPM is a good approach to implementing flexible business processes based on a more static set of underlying enterprise capabilities. And fourth, together, SOA and BPM provide a natural confluence of capabilities and processes. If you don't agree with these premises, perhaps you haven't been reading BPTrends that carefully, but read on anyhow.

Business and IT problems come in all shapes and sizes, so it should not come as a surprise that services also come in different shapes and sizes, as I have discussed in many of my previous articles. It is also a natural engineering approach to take a big problem and break it down into smaller, more manageable chunks. Again, this also applies to services. So now the question is: How do we assemble smaller services together to create bigger services?"

Well guess what; processes and composition also come in different shapes and sizes. The two main approaches are orchestration and choreography. Orchestration is where a central or master element controls all aspects of the process. Choreography is where each element of the process is autonomous and controls its own agenda. Think of a ballet performance. The music is orchestrated. The conductor provides master control over the orchestra, set's the pace and signals each instrument on when to join in. The dancing is choreographed. Each dancer knows their individual role, and executes it in response to the other dancers and at the tempo set by the music. The ballet emerges from the overall action of all the dancers, rather than from some central control. Also notice that a given enterprise may use both approaches, where each is most appropriate.

Orchestration

Orchestration is the most common approach, used both within service composition and business processes. With orchestration, you define the sequence of steps within a process, including conditions and exceptions, and then create a central controller to implement the sequence. Within an SOA, the individual steps of the sequence are implemented by operations on services. The sequence can be implemented with a variety of different techniques. Often, relatively simple service compositions are orchestrated in code, such as Java or C#, that resides inside the composite service. But for more complex orchestrations, you will often use a tool to create a visual model of the sequence, and then to generate the code that executes that sequence, typically within a dedicated run-time environment. This is the typical BPM approach that is well known within this forum.

Today's standards for orchestration include BPMN (business process modeling notation) for defining the visual representation of the sequence, and BPEL (business process execution language) as the 'code' that executes the sequence. Almost all SOA infrastructures provide some type of run-time BPEL engine, and most BPM products already support, or are in the process of supporting these standards in their modeling and run-time. In addition, BPEL is SOA friendly. BPEL is expressed in XML and defined by XML Schema metadata that is closely aligned with SOA standards. BPEL itself uses WSDL at two levels. First, WSDL defined Web Services are used to interact with capabilities required by the process. Second, every BPEL process is itself a Web Service described using WSDL

In summary, orchestration:

- Defines a single master controls of all aspects of a process (top-down approach)
- Supports a graphical view of the sequence
- Easily maps to SOA
- Is usually simpler to start with; but often harder to scale to more complex processes
- Is driven by the graphical sequence model, i.e. function follows form
- Represents the state-of-the-practice, and is supported by the majority of tools

So, the popularity of the approach is understandable. But it doesn't support all types of processes equally well.

Choreography

Choreography provides a different approach that is gaining acceptance in scenarios that have complex processes with many interacting parts, and event-based and agent-based systems. In a choreography approach, rules are created that determine the behavior of each individual participant in the process. The overall behavior of the process emerges based on the interaction of the individual pieces, each autonomously following their own rules. If you're familiar with the work going on with Complex Event Processing (CEP), you will see the similarity in the problem space of CEP and a choreography approach, i.e. how can you manage the interaction of multiple, independent events (or participants) to yield an overall, predictable business result.

There are currently two principle approaches to choreography, message-based, and work-component-based. The message approach is based on examining the messages between participants in an overall process. With this approach, you define behaviors by exhaustively capturing the message contracts between collaborating parties. This is the mechanism supported by the WS-CDL standard (Web Service Choreography Definition Language) and is often used for B2B applications. In B2B applications, which are by definition cross-enterprise, it is difficult to specify the implementation of specific participants, and there is no central authority for the overall flow. The message-based choreography approach is attractive because you only need to specify the message interchange definitions (syntax, semantics and behavior).

Another promising approach has to do with the configuration of process work-components. In this approach, you define the behavior of individual work-components and let the process behavior emerge as each specific process instance evolves. For example, you could implement routing behavior in individual work-components with a few simple rules such as: what capabilities a work-component needs to complete; what roles can meet each need; what causes a need to become active; and what causes a need to be considered complete. In a sophisticated system, these rules can be specified in the work-component's metadata, and then implemented in a work-component container. Readers familiar with agent-based systems should recognize the similarity here.

In summary, in choreography:

- The overall process behavior “emerges” from the working of its parts (bottom up). No global perspective is required
- Complex work processes are decomposed into work agendas where each autonomous element controls its own agenda
- Easily maps to event and agent based systems
- Is usually more difficult to start, but often easier to scale to complex processes
- Graphical representations can be derived from the process, i.e. form follows function
- Represents the state-of-the-art, and is gaining support with emerging tools

Tradeoffs

So, some useful guidelines for understanding which approach is most appropriate for a given scenario are:

Use orchestration:

- When off the shelf products are required (since this is the approach implemented by most current products)
- For composed services
- Where transaction semantics can be handled by compensation alone
- For relatively static process definitions
- Where a graphical process definition is desired

Use choreography:

- Where processes may scale to a very high number of component steps
- Where opacity of process details is desired among process partners (such as B2B)
- Where different process partners may require their own process customizations
- Where processes are highly dynamic or goal-seeking

Again, thanks to John Tibbetts for his insights and permission to reproduce the information. In my next column, I’ll turn the discussion to services, and look at the characteristics and design parameters of services that support these different approaches.