



## BPM and SOA

**Mike Rosen**  
Chief Scientist  
Wilton Consulting Group

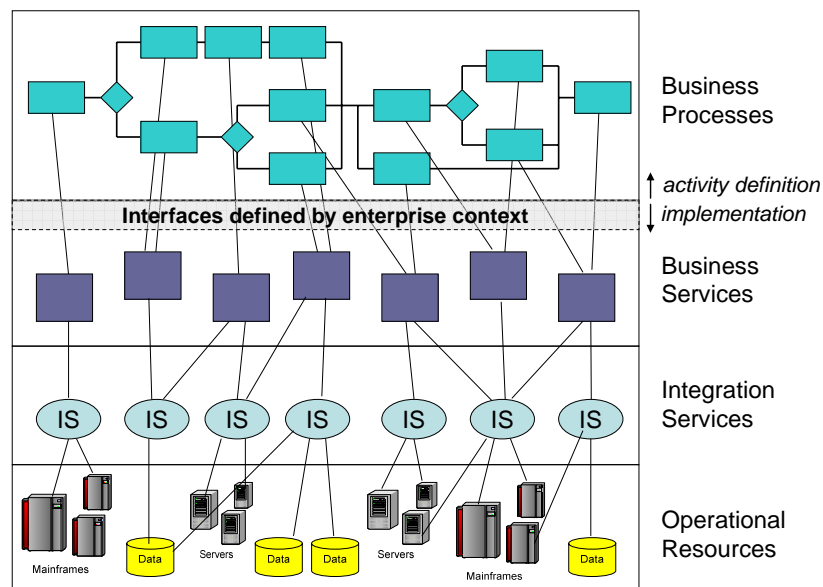
[Mike.rosen@wiltonconsultinggroup.com](mailto:Mike.rosen@wiltonconsultinggroup.com)

## Avoiding Common Failure Modes with BPM and SOA Projects

The idea that BPM and SOA work together is hardly new. In fact, I wrote about it for BPTrends four years ago. Since then, I've heard many people talk about it, but there still seems to be limited understanding about what it means from an organizational or implementation perspective. So, this month I'm going to review some of the key concepts of BPM / SOA and address some of the common failure modes that I see organizations get stuck in.

### Have an Architecture

First, let's review the relationship between BPM and SOA. Business Process Management (BPM) empowers a business analyst to align IT systems with strategic goals by creating well defined enterprise business processes, monitoring their performance, and optimizing for greater operational efficiencies. Each business process is modeled as a set of individual processing activities that are combined (composed) together to accomplish the overall business process. The individual activities are implemented as services (actually service operations) within the enterprise. Figure 1 shows the typical layered SOA architecture with business processes at the top.



**Figure 1. Layered BPM / SOA Architecture**

But, in order for the composition of services to actually result in meaningful business processes, the services that are composed together need to share a variety of important characteristics,

including a common understanding of the semantics of the new process. In addition, we have to face the realities of enterprise development. The set of services that will be composed into the enterprise processes are likely to be built by different organizations. So, the architecture has to put enough context and structure in place to make it possible for the different organizations to independently build services (according to their own business priorities) and still have the services work together at an enterprise process level.

To do so, SOA has to address all of the following issues: how services communicate at a technical level; how services are constructed; the different types of services and how they relate; how services are combined together (i.e., orchestrated); how services interoperate at a semantic level (i.e., how they share common meanings); how services support business processes; and how services contribute to IT and Business Strategy.

## Understand the Enterprise Context

Not understanding the enterprise context or not looking at things from a broader perspective is the most common failure mode that I see. Although BPM provides a wonderful abstraction for building business systems, all too often I see BPM being used to build bigger, better, siloed applications rather than contributing to an overall flexible, agile enterprise. This is where the service layer helps. SOA provides the layers that provide business capabilities and the bridge between the business processes and the operational resources, as shown in Figure 1. At the business process level, business services provide interfaces that directly support executing process activities. Importantly, those interfaces have been defined within an enterprise context to support consistency, interoperability, and reuse. At the operational resource level, SOA exposes existing capabilities as integration services. But it doesn't do this by directly mapping existing applications as services. Rather, it provides new service interfaces based on enterprise semantic and functional requirements and maps them to the existing systems. Finally, it joins these top and bottom layers together through service composition.

Unfortunately, just having well defined enterprise services isn't enough. We also need processes that support their entire lifecycle as illustrated in Figure 2.

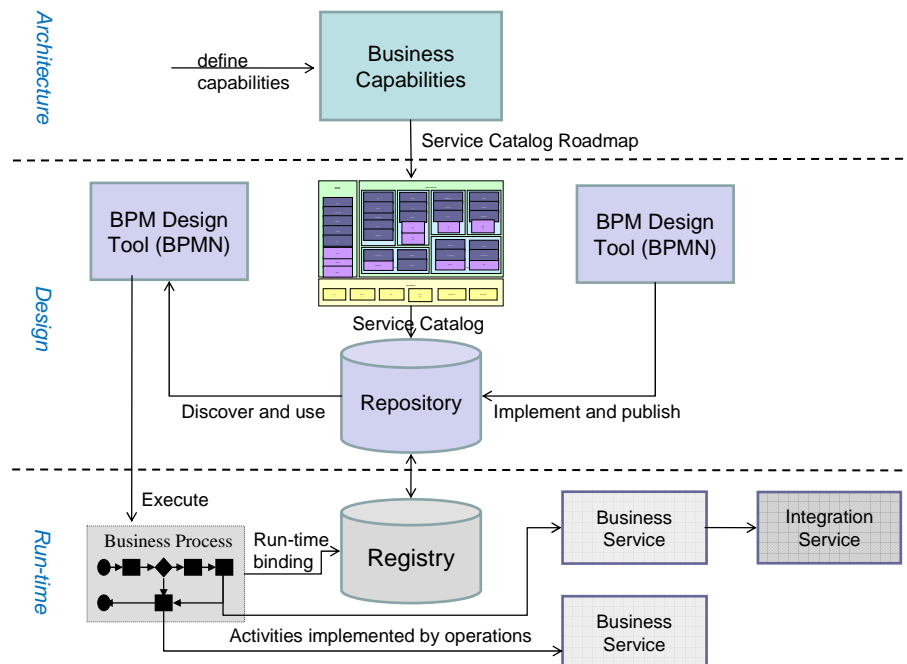


Figure 2. Business Service Lifecycle

The figure focuses the lifecycle of services around the concept of a Service Catalog (or service inventory). The lifecycle is described according to three sets of activities: architecture, design, and runtime.

At the architecture level we ask a set of business questions: “What are our business goals and strategy? What market position are we trying to obtain? What products and services do we need to achieve those goals and position? What information and business capabilities do we need to provide those products and services? And, finally, what SOA services will we need to provide those capabilities? The answer to this final question is translated into an organized catalog of services (illustrated by the colored grid) and a roadmap for implementing them over time.

At the design level, service development teams implement the services according to the roadmap (more on this later). Meanwhile, business analysts design their business processes. Here is a critical integration point. The service repository (design time) must integrate into the BPM modeling tool so that business analysts can easily discover and use services defined in the service catalog according to the business architecture. This requires two things. First, a technical integration where the BPM tool can physically read the repository and provide relevant information to the designer. Secondly, a development method that takes the enterprise context into account when designing the business process. The list below describes a typical BPM design sequence.

- Identify and analyze the use cases (scenarios)
- Break the scenarios down into activities
- Look for services that perform the specified activities
- Use existing services where possible, otherwise
- Design and implement new services

Seems okay? Well it's normal, but it's not okay. Look at the slightly modified sequence below, paying attention to the highlighted steps.

- Identify and analyze the use cases (scenarios)
- **Understand what services are defined in the catalog and their responsibilities**
- **Using existing services to frame the design**, break the scenarios down into activities
- Use existing services where possible, otherwise
- Design and implement new services

The change in sequence is minor, but it makes all the difference in the world. Think about the enterprise context first and design your business process to fit within it, rather than designing your process in a silo, and then hoping that some services might fit.

Returning to figure 2, during BPM design, we look at the service catalog and use it to provide context for the process design. Once the process is implemented and tested, we can go to the runtime level.

At runtime, the business process is executed by the BPMS. As each activity is executed, an operation on a business service will be invoked. First, the infrastructure looks up the services in the registry to discover their location and bind to them. Then, the registry has done its job and is out of the way, and process activity execution calls directly to the service interfaces. So, why did I bother to include the registry in this picture at all? Because another major failure mode is confusing a registry with a repository.

A repository is used during design time to search for, discover, learn about, and evaluate services. A registry is used at runtime to look up a specific service name and find its network

location. Obviously, there is some overlap between these two things, but they are also very different. Trying to use a registry to support design time discovery is a recipe for failure.

## Be Agile

Finally, be agile about how you develop services (and processes, for that matter). Start small, deliver value, and learn what processes and governance work in your organization, then develop more services, deliver more value, and perhaps expand the development of services to other groups.

In a recent research report that I wrote on SOA case studies, one of the key findings was that successful implementations have an architectural vision, but that they start small and focus on delivering business value. Then they examine lessons learned, factor them into their next set of services, processes, policies, and governance, and build more services.

Another important aspect of being Agile is not trying to build the ultimate reusable service from day one. Instead, when we build Version 1 of a service, we do it to meet the requirements of a specific business process or project (but, of course, in compliance with the business model, enterprise semantics, and service catalog). Then, when it's time for another process to use the service, we update it to provide the additional features or information that process needs. No problem. That's what versioning is all about. Voila, now we have Version 2 of the service. Notice that this method doesn't require the first user of a service to pay for the whole service, only for the parts they need. As more features are added, the work is funded by the projects that need them. I've seen some services that have evolved to 20 or more versions, although, as more and more processes use the same service, it tends to reach an equilibrium where few if any enhancements are required to support additional user processes. At any given time, two or three different versions of the service will be supported simultaneously. Eventually, when the business processes that use Version 1 of the service need to be updated, they will also be updated to use the latest version of the service. This might sound a little difficult, and, for sure, there is some complexity with it. But, it is far easier and more practical than trying to build the ultimate service from the beginning, much less figuring out how to fund it.

Together, BPM and SOA provide a perfect combination for enterprise computing. BPM provides the higher-level abstraction for defining businesses processes. Services provide the capabilities that support those processes across the enterprise. But, to get the benefits of agility and flexibility requires that the broader enterprise context be considered as part of an agile, architectural based approach to process and service design.