A BPTrends Column May 2009

BPM and SOA

Mike Rosen
Chief Scientist
Wilton Consulting Group.

Mike.rosen@wiltonconsultinggroup.com

Designing Service Oriented Solutions
Part I – The Business Model

Business / IT alignment has been an elusive and consistent goal of both IT and the business. But, one big challenge IT faces is what to align with. Often, the business goals and strategy are not well understood or articulated. In this case, business architecture (BA) can provide important clarity. However, like anything else, BA in isolation will not provide any better alignment. Rather, the BA has to be integrated into the overall analysis and design process. This two part series describes the role of BA in designing service oriented solutions. Part I provides an overview of the important aspects: process design, information design, and service identification. Part II provides an overview of the overall design process.

The foundation of a business aligned SOA implementation is an enterprise business model, containing the primary representation of the resources (business, IT, data, etc) and processes involved in meeting the enterprise operational, tactical, and strategic business goals. Business architecture is an essential component of a successful service-oriented solution. In particular, BA must answer the following questions:

- What business are you in?
- What are the goals and objectives of your particular business?
- What outcomes are needed to achieve those goals?
- What is the strategy for achieving them?
- How will they be measured?
- What capabilities and information are needed to achieve those outcomes?
- What processes, services, information, and rules are needed to implement those capabilities?
- What existing applications provide basic capabilities and information?
- How are the applications, processes, etc. aligned with the business strategies and goals?

Business architecture helps us understand and answer these questions and describes how to provide traceability from the operational concepts of processes and services through to the concepts of tactics and objectives all the way up to business goals and strategy.

Business Processes

Business tactics and objectives are typically defined for particular business processes. A business process is a group of logically related (and typically sequenced) activities that use the resources of the organization to provide defined results. Business processes deliver value in the form of product or service, either internally, or to an external party such as a customer or partner.

In order to accommodate the needs of both executive management and business process owners, business processes are typically defined at two levels of detail: "One model, for the

executives, contains a set of high-level business scenarios that show the intent and purpose of the organization. The other model, for the business process owners, contains a detailed set of use cases that define how the organization needs to function internally. For each high-level business scenario, you could define one, or several, detailed business use cases representing the same activities in the organization..." (IBM's Rational Unified Process (RUP) for SOMA).

This kind of analysis can be thought of as a type of process decomposition. The high level scenarios provide the high-level description of what business systems do. This level of processes defines only the highest level enterprise scenarios and is rarely detailed beyond the narrative. Processes such as 'order to payment' fit this level. These descriptions serve as the starting point for process decomposition (sometimes called Level 2 processes), which are the foundation of the enterprise business model. 'receive purchase order' is an example of a process that supports the order to payment scenario.

Level 2 processes are also a foundation for the definition of the process task or activities which are used to define the high-level business services. For example, the receive purchase order process might be composed of purchase order, customer, inventory, credit checking and other business services. In other words, business process decomposition provides three levels of hierarchy – top level scenarios, made up of level 2 processes, composed from business services.

The goal of enterprise SOA is to expose an organization's computing assets as reusable business services, exposing basic business capabilities and information, which can be used and integrated more readily using business processes. The relationship between business services and business processes (Figure 1) paves the way to a truly flexible enterprise:

- Business services support stable business artifacts, incorporating processing and rules whose interfaces change fairly rarely. On the other hand, service implementations can and typically do change frequently.
- Business processes support fluid business procedures and rules, which can change every few months or even weeks.
- The interaction between business processes and business services is based on enterprise semantics, which minimizes the impact of services changes on the business processes and simplifies constructing processes from business services.

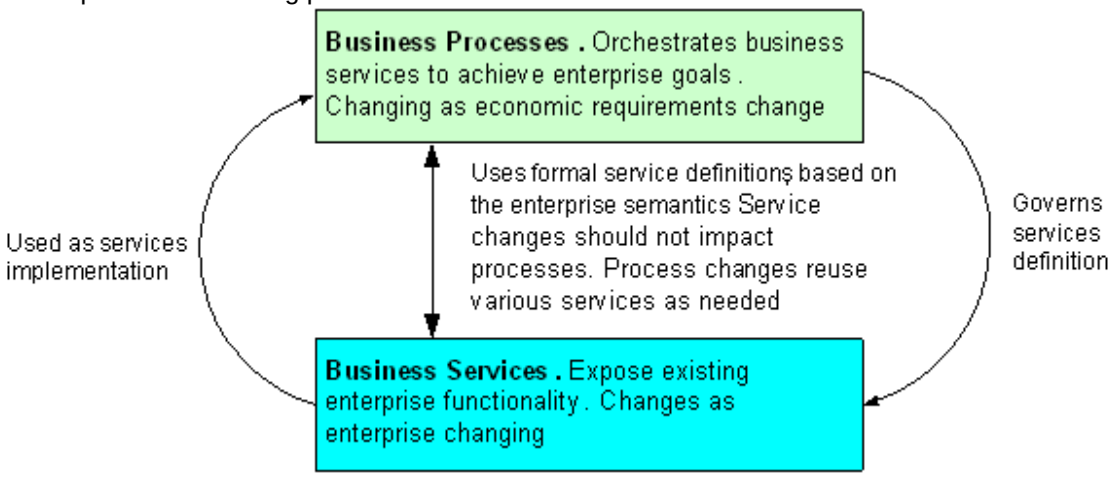

Figure 1 – Process / Service Interaction

This separation of responsibilities enables business analysts and IT architects to reuse IT capabilities, encapsulated in business services, through the composition of business processes. This simplifies the creation of the new processes and optimization of existing ones. More importantly, once designed, processes can be quickly modified in response to market conditions.

All this translates into increased business flexibility and competitiveness while reducing the incremental costs of making frequent process changes.

Information Design

The next step in the process definition is identification of the enterprise semantics (semantic information model) — a definition of the standard business entities for the enterprise; for example, insurance policy, claim, etc. A common semantic definition ensures that:

- Each term throughout the enterprise has a clear and concise definition.
- All enterprise terms are used consistently (mean the same thing and use the same definitions) throughout the enterprise.
- Each term is used in at least one process/activity definition.
- Only terms defined in the enterprise semantic information model are used by process/activity definitions.

The semantic information model is influenced by both the business architecture and the information architecture. The business architecture identifies the processes required to support the business goals and objectives. The semantic information model defines the information, concepts and meanings that must be common throughout those processes to effectively pass information between the process tasks. This corresponds to the information architecture concepts of illustrated in figure 2

.

The semantic data is not the same as the domain data. It does not define all of the details of the information needed within each step of a process. Rather, it defines the information that must be common between then. Each individual processes step (implemented by a business service) will provide any transformation required between the semantic information model and their own internal domain model.

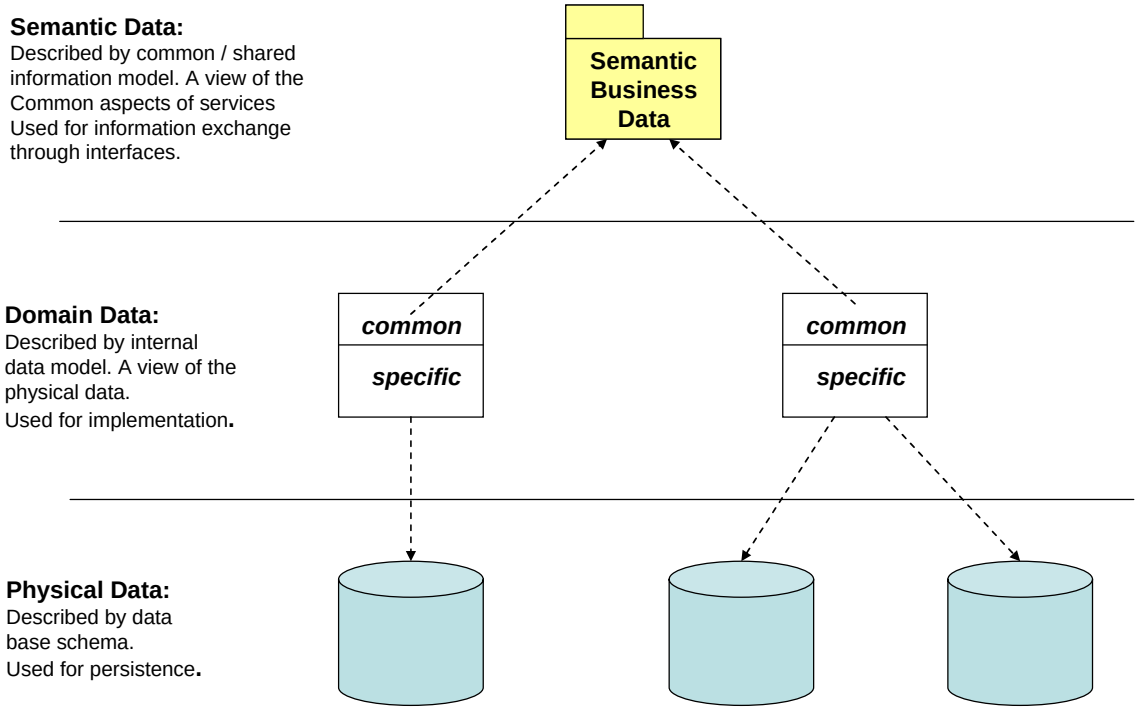

Figure 2 – Semantic Information Model

Although the semantic information model seems similar to a standardized enterprise data model, the two are very different and should not be confused with each other. The semantic information model defines the messages exchanged between services through their interfaces. The messages implement inter-service communication. As such they are transient and do not reside in a data store (at least not explicitly). In contrast, the enterprise data model defines the data structures and relationship between data in the databases. In practice, implementation of a Service-Oriented Architecture involves service enabling existing enterprise applications. However, changing the underlying data model is an extremely difficult proposition that often requires complete application rewrites. In other word, it's probably not going to happen, so a system that provides interoperability without changing existing models is much more likely to succeed.

An SOA implemented based on a semantic information model provides a semantically interoperable SOA providing enhanced interoperability between services. At the interface level, all of the services work with the same objects. In effect this eliminates the necessity of message transformations between services. Because service interfaces are created based on the standard enterprise semantic information model, it is guaranteed that every service will "understand" and correctly "interpret" any message, regardless of who the service consumer is.

There are plenty of industry organizations today defining semantics for a particular industry, such as ACORD for insurance, or HL7 for healthcare. Their semantic dictionaries should be considered a starting point for the creation of any enterprise semantic information models.

Services Identification

One of the most important tasks during implementation of an SOA solution is the proper definition of business services, based on the decomposition of the problem domain (level 2 process definitions). Sometimes, it seems like the simplest approach to decomposition, and consequently service definition, is to directly expose existing application's functionality as a set of services (decomposition based on the existing application portfolio) – similar to a traditional Enterprise Application Integration (EAI) practices.

Unfortunately, this approach rarely works. It "is in essence a technology first approach and is a recipe for disaster and/or serious over-engineering" (Grady Booch. Software architecture, software engineering, and Renaissance Jazz blog : SOA Best Practices. March 11, 2006. - 03.ibm.com/developerworks/blogs/page/gradybooch?entry=soa_best_practices). A better approach is based on the decomposition of the enterprise-wide business model – designing a set of services that implement the business architecture supporting the current business goals of the enterprise and providing capabilities for future changes. It requires you "to start with the scenarios/business needs, play those out against the existing/new systems, zero in on the points of tangency, and there plant a flag for harvesting a meaningful service" (Grady Booch).

This approach leads to the creation of business-aligned IT services, available to participants throughout the enterprise across multiple lines of business or even outside of the enterprise, that collectively fulfill an organization's business processes and goals.

Hierarchical decomposition, based on the enterprise business model is typically not sufficient for proper services identification. Although it provides an alignment between business and IT, it does not guarantee that the resulting services will adhere to the basic service tenets. Fundamental service characteristics (autonomy, loose coupling, etc.) also need to be considered in the design process. But, still this is not enough. The services need to be defined within the context of the overall enterprise. To do this, we need to think about the way we design systems and decomposition differently. To overuse a phrase, we need a paradigm shift in design practice.
For example, a typical approach to SOA design might incorporate this sequence:

For each business domain, identify and analyze the processes.
Break the processes down into tasks that are implemented by services.

Look for existing services that perform the specified tasks.
Use existing services where possible.
Design and implement new services.

This probably seems like a pretty reasonable approach, but let's look at an SOA focused sequence and compare:

For each business domain, identify and analyze the processes.
Understand what services currently exist (or are planned) and their responsibilities.
Using existing services to frame the design, break the process down into tasks that are implemented by services.
Use existing services where possible.
Design and implement new services where necessary.

The difference comes at the breakdown of processes into tasks and services. The difference may seem subtle, but the effect is huge. In the first approach, we are free to come up with almost any reasonable sequence of tasks to implement our process. There could be dozens of possibilities. Then, we look for existing services that do things our way, but probably don't find very many. Instead, we implement new services, but ones that overlap with existing services.

In the SOA approach, we factor in the existing services first, and then design around them. They provide a design constraint that limits the possible solutions to a few, instead of dozens. Now, when we go to use existing services, they've already been designed in, and work with our new solutions, and support our enterprise. Instead of creating new services, we've facilitated reusing existing ones.

The crux is this. We are not designing a solution or process from scratch. Instead, we are starting with an existing base, and building our solution on top of it. We are extending and reusing, adding value to what exists, not duplicating responsibilities and adding inconsistencies. But to make this work, we need an organization of services that makes it easy to understand the overall set of services. We call the overall set of services the 'service inventory'.

The service inventory lays out the overall set of services and their relationships to each other and the overall enterprise goals. We can think of the service inventory as a 'responsibility map' of service interfaces. It should clearly describe the overall set of services, and what responsibilities the different service groups perform, and don't perform. The service inventory helps in two important service design activities.

First, the inventory allows us to quickly scan the overall set of services at a high level and then to dig deeper into groups of services within a given area. This helps to locate the services to support our 'look first, design later' approach.

But at least as important, the inventory helps us make decisions about what functions to include within our service implementations, and what functions we should expect to be performed by another service. If we need to implement a new service, we have to make sure that it doesn't duplicate functions that are already (or plan to be) implemented by other services. This is where the responsibility map aspect of the inventory is important. It must clearly define the boundaries of responsibility for services and service groups.

Looking Ahead

In this Column, we have examined the relationship of business processes to business services in terms of functional decomposition. Equally important to a service is the information that it acts on, and we have also examined the role of a semantic information model to define the messages passed through service interfaces. Finally, we have examined a 'look first' approach to service identification aimed at maximizing reuse and minimizing redundancy. In Part II of the series, we'll

look at how these concepts fit into the overall design process. These topics are based on information contained in the book “Applied SOA: Service Oriented Architecture and Design Strategies” published by Wiley in June 2008.