Process-oriented modeling for SOA, Part 1: A technique for process decomposition

How to specify process models that align business processes with SOA

Ruud Schoonderwoerd

November 04, 2008

This first article explores decomposing a business process into different layers of responsibility— as opposed to different levels of detail—and also looks into the role of the process controller as well as how services are identified by where they are needed. In this series, learn about a new decomposition technique that can help you specify business processes that are aligned with a Service-Oriented Architecture (SOA).

View more content in this series

Introduction

Process modeling is increasingly important for IT projects based on SOA. You can use process modeling to identify services, structure requirements, and define business processes that are executed on a business process management (BPM) platform.

Part 2 will expand on information in this article by using a set of business process patterns. Future articles will explore how the approach has the potential to improve how business analysts gather requirements and structure use cases, and how architects specify SOA solutions.

Process modeling, which is understood by both business and IT stakeholders, is a necessary tool for delivering the mutual benefits of SOA and BPM: better alignment between business and IT and greater re-use.

In this series, learn about process-driven modeling of SOA solutions. This article explores a process-decomposition technique that lets you specify process models which are closely aligned to an SOA-based target architecture, so your process models show better fidelity to the solution being implemented.

Why we need SOA-aligned process models

Delivering the benefits of modeling is still difficult because most process modeling techniques are based on concepts that predate SOA and BPM. Traditionally, different process models are
produced within an enterprise for different purposes. For example, process modeling may be used in strategy definition, organizational change, requirements solicitation, operational optimization, functional specification, user interface flow, work instructions, and enterprise application integration (EAI) flows. Even if two models are meant to represent the same process, if they're produced for different purposes, they are likely to be different.

Process modeling has maintenance, traceability, and testing issues:

- When a process model evolves during the life cycle of an IT project, it often originated from an ambiguous model initially produced for a purpose other than what it's used for later in the cycle.
- To avoid the aforementioned issue, a brand new model is sometimes created at each stage of the life cycle, creating the need for explicit traceability artifacts, which are expensive to maintain.
- Often, practitioners in the upstream (modeling) phases of an SOA project don't understand the impact of their work downstream.

As a result, process models developed in the earlier phases don't have a clear link to the actual solution being implemented. Or, the models drive the solution in the wrong direction.

SOA-aligned process models could be produced using other, less precise, process models as input. A direct approach to specifying solutions has many benefits:

**Facilitate handover**
- Improved handover of process models from business analysts to architects—and from architects to designers, developers, and test teams

**Reduce refactoring**
- Decreased refactoring of models from one life cycle stage to another

**Enhance traceability**
- Better trail of BPM and SOA-based projects

**Meet client expectations**
- Improved rate for meeting client expectations by visualizing the solution, improving clarity of communication, and improving alignment of what is being requested with what will be delivered

**Increase speed of project delivery**
- Using guidelines and patterns leads to simplified and more accurate designs

**Define a common language and framework**
- Assists in exchange of process models between practitioners

**More efficient testing and verification**
- A better link between requirements and what was delivered yields efficient testing and verification

**Increase potential for re-use**
- Reused services are more likely to have a clearly identifiable artifact in other life cycle stages

### The IBM SOA reference architecture

The process modeling technique is illustrated using the IBM® SOA reference architecture, which is a blueprint used for many SOA initiatives by IBM customers. [Resources has more detail about the
IBM SOA reference architecture.) This article focuses on the three layers within the architecture that are most relevant to the process modeler: the consumer layer, the business process layer, and the services layer.

**Services or processes?**

Naming of reference architecture layers can sometimes be misinterpreted:

- "Business processes" suggests that this is where business processes are executed. In practice, there are consumer-specific business processes in the consumer layer too, as well as manual business processes that have no systems implementation at all. A more accurate but longer name would be "orchestrated business processes," because these processes are often executed using an orchestration engine that's part of a BPM platform.
- "Services" might imply that all services belong in this layer. But business processes in the business process layer are typically exposed as services, too. Even consumer processes may be exposed as services in a business-to-business (B2B) interface.

This article might refer to business processes that are not in the business process layer, or services that are not in the services layer. Where relevant, I'll explicitly refer to the layer.

**Consumer layer**

The consumer layer contains the applications used by end users within an organization, or the inbound interfaces to external organizations (such as customers or partners). Consumer applications are responsible for maintaining dialogues or sessions with users. The consumer layer is sometimes called the presentation layer.

**Business process layer**

Processes in the business process layer are invoked (using services) by applications in the consumer layer. They achieve specific business objectives and are composed of multiple activities, which may in turn use services provided by the services layer. Processes in the business process layer are often implemented as orchestrated processes executed on a BPM platform. For the technique in this article, it's necessary to distinguish between long-running and short-running processes:

- Long-running processes, also known as workflow processes, have at least one step that involves: a person performing an activity, a wait for an external event to happen, or processing that is prohibitively long. A consumer application cannot expect an immediate result from a long-running process.
- Short-running processes are the composition of services that are fully automated and completed (almost) instantly. A consumer application can expect a result almost instantly when invoking a short running process.

There are also other good architectural reasons to make this distinction.

**Services layer**

The services layer provides building blocks in the form of services to the business process layer and consumer layer. These services, which are automated, may be represented as activities
in a business process. Figure 1 highlights the layers of the SOA reference architecture that are relevant to this article.

**Figure 1. IBM SOA reference architecture**

![IBM SOA reference architecture diagram](image)

The technique described later in this article requires that process models are layered in alignment with these layers. Before going into more detail, though, the next section discusses human activities that are part of long-running (workflow) processes.

**Human activities in long-running processes**

A step in a long-running process can correspond to a human activity (a manual workflow item). Typically, such tasks are performed by an employee of the enterprise who is fulfilling a particular role in the process. BPM platforms may provide a way of sending this activity to a task list, and present a user interface (UI) for users to select tasks from the list. The task list itself isn't a feature in process models; it's abstracted away.

**The layering principle**

A key feature of SOA is the layered architecture that follows the **layering principle**. It is decomposed into several layers, where each layer serves the layers above, to provide a set of services without revealing how they are implemented. In turn, layers rely on the services provided by the underlying layers. Layering is extremely powerful in tackling complexity. Layered systems are easier to understand; they abstract (shield) implementation details of underlying functions from the layer being considered.

Most successful, long-lived, complex computing platforms incorporate layering. The most famous example of layering is the **Internet Protocol stack**.

**Violating the layering principle**

When a layer calls a layer above, the layering principle is broken. This should be avoided for many reasons. For example, it will make the design less traceable, may cause unintended recursion, and leads to losing the benefits of layering.
When you are deciding how to represent human tasks in process models, there appear to be three possible options:

- **Option 1:** From a process modeling point of view, human activities are steps in a business process similar to automated activities in the services layer. They should therefore be modeled as such and be logically considered in the same layer as automated activities.
- **Option 2:** From a technical point of view, human activities require a user interface that is subject to the same access control mechanisms and other capabilities as other consuming applications. Human activities can also consume services provided by the underlying layers. Their implementation should be part of the consumer layer.
- **Option 3:** Human activities are part of the business process layer itself, as they are provided and mediated by a BPM solution.

These views appear to be incompatible, and the first two options break the layering principle (see sidebar), each in different ways:

- Human activities may need to invoke short-running business processes, which would be in a higher layer if human activities were considered part of the services layer.
- Business process modelers would think of human process activities being invoked by business processes. *(Invoked is used in a business rather than technical sense, as the invocation mechanism is effectively the allocation of a task to the user’s task queue.*) If human process activities were part of the consumer layer, from a process modeler view this would effectively mean that the business process layer invokes the layer above.

The solution is a refinement of the third option, where human activities are placed within the business process layer, between long-running and short-running processes. In doing this, you can also ensure that human activities will not invoke the long-running processes layer.

**SOA-aligned process decomposition**

For SOA projects with a BPM component, you must adhere to the process-decomposition structure shown in Figure 2. *(This framework drives the rationale behind the patterns in Part 2 of this series.)*
Process models developed to specify functions of an SOA-based solution should follow the decomposition structure in Figure 2. Each individual process belongs to one of the layers.

An enterprise service bus (ESB) can present service interfaces to different layers (long-running process, short-running process, human activity, automated activity), but these should not be treated as equivalents. The framework’s logical structure imposes rules on what should be allowed to call what, so models and designs are straightforward to track and understand.

Processes can only call or subscribe to events from processes in downstream layers, as indicated by the arrows. They should also be agnostic of the processes in the upstream layers. Each layer is discussed next.

**Manual processes and customer experience processes**

The manual processes and customer experience processes layer sets the context for processes in the lower layers. The processes in this layer have no associated layer in the SOA reference architecture; however, they help explain what leads to the launch of a consumer process.

**Consumer processes**

Consumer-specific process models specify the process of end user applications. They show how such an application interacts with the underlying processes. The consumer layer application holds the session state required to maintain dialogues with the user. Users could be external users (customers), internal users (employees), or system users (scheduled batch jobs or B2B interfaces). The main purpose of the consumer layer is to gather all the information required to be able to start a business process or transaction.

Any logic that is defined within the consumer processes must be specific to the application (or channel). Therefore, it should remain distinctly separate from the business processes layer.
Consumer-specific processes call long-running processes in fire-and-forget (or request-acknowledgment) mode. They will not wait for a response, but they can also call short-running processes using a request-response interaction. Consumers need to know whether the business process being invoked is long-running or short-running, as they can expect only an imminent result from the latter.

Business Process Modeling Notation (BPMN) is fast becoming the de facto notation standard for process models. However, the artifacts used to represent consumer-specific processes do not necessarily need to be BPMN models. (It's hard to include visual representations of UIs in BPMN diagrams.) Alternative artifacts could be use cases, screenflows, wireframes, or UI prototypes. In such a case, a specification must make it clear where underlying business processes (short- or long-running) and services are invoked, and how results are handled.

**Long-running processes**

Long-running processes (or workflow processes) are invoked by consumer processes, or as a result of events generated by short-running processes (see Event-based SOA). They are intended to be reusable across these two types of processes.

Long-running processes are nonconversational, from the point of view of the consumer process that invokes them, and they may call short-running processes or automated process activities. They may also "invoke" human tasks by sending a task to a worker's work list or queue.

**Human activities**

From a process modeling viewpoint, and within this context, human activities are invoked by long-running processes. Their execution typically starts when a person who has been assigned the activity selects it from a work list. Human activities can themselves be represented as processes, as it may take a number of steps (screens) to complete them. Human activities may invoke short-running processes and automated activities.

Technically, human activities are started as a worker logs in to a work application and selects a task (or searches for the process instance). In this framework, however, the work application is considered a kind of invocation mechanism. From a process modeling point of view, human activities are invoked by the long-running process they are part of. This abstraction becomes even more valid as modern BPM platforms provide the UI aspect of human tasks, including a work list or queue UI. A work list can therefore be considered an abstract entity that does not require modeling.

**Short-running processes**

Short-running processes are invoked by consumer processes, long-running processes, or human tasks. They must always be nonconversational and typically orchestrate automated process activities. Short-running processes have the capability to inform the caller of completion (possibly including result data) in real time.
Automated activities

Automated activities are atomic steps within business processes. They might represent validations, create/read/update/delete data operations, business rules, and calls to external systems.

Automated activities correspond to the services layer in the SOA reference architecture. During a process modeling stage, activities could generally be considered as candidate service operations.

The process controller

The approach in this article incorporates the notion that there should only ever be one entity—the process controller—that controls a given process.

A business process flow is always controlled by something. Each process, or subprocess, should only ever have one process controller. If you cannot identify this, or if a newly modeled process jumps from one controller to the next, something must have gone wrong in the decomposition. You likely identified another subprocess.

Examples of process controllers are shown in the following table.

<table>
<thead>
<tr>
<th>Process controller</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A person</td>
<td>A person manages the flow of the process.</td>
</tr>
<tr>
<td>A piece of paper</td>
<td>The paper that flows through a process has information on it that identifies what needs to be done next (possibly in combination with work instructions).</td>
</tr>
<tr>
<td>Instructions</td>
<td>There is a separate set of instructions, either explicitly written down or in people's heads, that guides the process.</td>
</tr>
<tr>
<td>Screenflow</td>
<td>A person is entering information on a screen and is guided to the next screen by the underlying screenflow mechanism.</td>
</tr>
<tr>
<td>System orchestrated short-running processes</td>
<td>A system orchestrates a series of fully automated activities.</td>
</tr>
<tr>
<td>System orchestrated long-running processes</td>
<td>A BPM system orchestrates a series of activities. Human tasks are sent to a work list, and system operations, such as service invocations, are performed automatically.</td>
</tr>
</tbody>
</table>

In the process-decomposition framework, each controller belongs to one of the layers. The first three examples in the table above correspond to the manual processes and customer experience processes layer. Screenflow could be used in the consumer layer and human activity layer. For the other two examples, the corresponding layer is (hopefully) obvious.

The process controller concept forces the process modeler to acknowledge the mechanism by which the flow in a process is controlled. It will also:

- Avoid difficult questions later from IT teams who need to implement aspects of these processes.
- Facilitate transition to an SOA-based architecture, as each layer in the architecture corresponds to a different process controller.
Though a process controller is an abstract notion, it can be represented graphically, as shown in Figure 3.

**Figure 3. A process and its representation with an explicit process controller**

![Diagram](ibm.com/developerWorks/developerWorks®)

Each time an activity in the process completes, control is handed to the process controller. This "role" has an internal representation of the process flow and determines the next step based on the representation. The notation above is never actually used, because it renders a process model unreadable.

**Event-based SOA**

Some scenarios include a long-running process that needs to be invoked by a short-running process, which would result in a violation of the layering principle. This typically occurs when the long-running process represents the fulfillment of a transaction that's represented by a short-running process. For example, in an order management scenario the transaction Place Order (short-running) needs to result in the processes Fulfill Order and Replenish Stock, which are both long-running.

The main purpose of the layering principle is for lower layers to be unaware of the higher layers they are serving. Typically it would mean that a higher layer can call a lower layer, but not the other way around. It could also mean that a higher layer listens and acts upon events sent by a lower layer.

For example, the short-running process Place Order may generate the event New Order Arrived. An event listener in the long-running processes layer may receive this event and launch the Fulfill Order and Replenish Stock processes—without the Place Order process being aware of the existence of these processes.
The link to use-case modeling

Business process models that are designed using decomposition techniques and patterns are very suitable for identifying and specifying use cases (and the "include" and "extend" links between them) and services. Stay tuned to this series for more on the topic.

Conclusion

Unlike most established process modeling techniques, the technique in this article does not use a process hierarchy. "Hierarchy" implies a tree structure, where each process belongs to a higher level process (except for processes at the highest level). In such a hierarchy, subprocesses tend to be embedded, as opposed to independent, which is detrimental to achieving re-use.

Think "stack" instead of hierarchy.

The technique described in this article uses the concept of a process stack, where a higher layer relies on the services provided by a lower layer. The services may have an underlying process implementation. These services and their incarnations as processes are independent and can therefore be used across different higher layer processes.

Similarly, the notion of "level of detail" is often used to help identify subprocesses. The technique in this article considers layers of responsibility instead. A higher layer of responsibility is not necessarily concerned with a higher level of detail. For example, you could argue that the level of detail in a consumer process is the same as in a long- or short-running process. However, the processes within those layers have different types of responsibilities; one always calls the other, and not the other way around.

Let service identification be consumer-driven

With the process-decomposition technique, models of consumer processes identify candidate services you need in the business process layer, and models of long-running and short-running processes identify candidate services you need in the services layer. Services are thus identified by where they are needed, and where they are used. A future article in this series will explore service identification and specification in more detail.

Use existing systems analysis

The analysis of existing systems should provide the bottom-up view. Where possible, process models should align with the analysis by selecting the automated activity steps within processes to correspond to services that can be offered by an organization's existing applications.

Acknowledgments

Ideas don't occur in isolation. I am grateful to all IBM colleagues I have had discussions with. In particular I'd like to thank: Mano Cheema, Sunny Shah, Mike Evans, Bruce Anderson, Joe Hardman, Simon Plackett, Kim Clark, Richard Solly, Pete Cripps, Ian Turton, Tony Carthy, Ali Arsanjani, and Doug McDavid for their input.