Modeling SOA: Part 1. Service identification

Jim Amsden

October 02, 2007

This article is the first in a series of five articles about developing software based on service-oriented architecture (SOA). It shows how to use UML models extended with the IBM® Software Service Profile to design an SOA solution that is connected to business requirements, yet independent of the solution implementation. The author describes the business goals and objectives and the business processes implemented to meet those objectives, and then explains how to use the processes to identify business-relevant services necessary to fulfill the requirements that they represent.

View more content in this series

How modeling improves SOA

The power of an SOA is in its ability to enable business agility through business process integration and reuse. SOA achieves this in two ways: By encouraging solutions organized around reusable services that encapsulate functional capabilities separated from their implementations and by providing facilities for managing coupling between functional capabilities. Modeling can be used to bridge the gap between business requirements and a deployed services-based solution. SOA models raise the level of abstraction to allow you to focus on business services. Model-driven development approaches can be used to generate SOA implementations, by using platforms such as Java™ 2 Platform, Enterprise Edition (J2EE) or IBM® CICS® that meet business functional and nonfunctional objectives while enabling business agility.

The term service-oriented architecture (SOA) has several connotations. Practitioners commonly use SOA both to define an architectural style and to describe a common IT infrastructure that enables IT systems that are built using that architectural style to operate. These are useful technology-focused perspectives, but, by themselves, they are not enough.

To achieve its potential, an SOA-based IT infrastructure (hereafter, referred to as simply SOA) needs to be business-relevant, thus driven by the business and implemented to support the business. We need a way of designing SOA solutions that are connected to the business requirements that they fulfill. This is hard to do if the business requirements are given as a simple list of requirement items and the level of abstraction of the SOA is captured in several XML documents that describe a collection of Web services.
What we need is a way to formalize business requirements and raise the level of abstraction so that SOA can more closely resemble business services and how those services might meet business goals and objectives. This ties the deployed solution to its intended business value. At the same time, we need a way of isolating business concerns from the evolving SOA platforms that support them.

Modeling and model-driven development (MDD) can help achieve these goals. Models allow us to abstract away the implementation details and focus on the issues that drive architectural decisions. To some extent, the approach we will be describing applies one of the fundamental principles of SOA to the development of SOA solutions: separation of concerns and loose coupling. Here, we cleanly separate the tasks and responsibilities of business analysts from those of IT staff members.

Usually, business analysts will be focused on business organizational and operational requirements necessary to meet business goals and objectives that achieve some business vision. Often, they are not concerned with (nor sufficiently skilled to deal with) IT concerns, such as reuse, cohesion and coupling, distribution, security, persistence, data integrity, concurrency, failure recovery, and so forth. Further, business process modeling tools don't often have the capabilities necessary to address these concerns, and, if they did, they probably wouldn't be effective tools for business analysts.

**About this series on modeling SOA**

This series of articles shows how to use UML models extended with the IBM® Software Service Profile to design an SOA solution that is connected to business requirements, yet independent of the solution implementation. It is generally easier to understand concepts by following a concrete, typical, and complete example. That is the approach we'll take here. We won't spend much time dealing with UML details but will, instead, focus on how you can use UML to help with design and development.

Although this series of articles is about Web services solutions from SOA models, we start here by focusing on the business goals and objectives that we're trying to achieve, so that we ground our solution in achieving something of value to the business. We then explore a business process that models what has to be done in the business to achieve those goals and objectives. This will provide the business functional requirements that the SOA solution has to meet. We'll then use this business process to help identify services that will be useful in creating an SOA solution that meets the business requirements.

In follow-on articles in this series, we'll create service specifications and implementations that fulfill those requirements with an architecture that enables future reuse and business agility. Finally, we'll use MDD to generate a Web services solution that can be deployed and executed.

To make the example even more real, we'll use existing IBM® Rational® and IBM® WebSphere® tools to build the solution artifacts and link them together, so that we can verify the solution against the requirements and more effectively manage change. Table 1 provides a summary of the overall process that we'll use in developing the example and the tools used to build the artifacts.
Table 1. Development process roles, tasks, and tools

<table>
<thead>
<tr>
<th>Role</th>
<th>Task</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Executive</td>
<td>Convey business goals and objectives</td>
<td>Rational RequisitePro</td>
</tr>
<tr>
<td>Business Analyst</td>
<td>Analyze business requirements</td>
<td>WebSphere Business Modeler</td>
</tr>
<tr>
<td>Software Architect</td>
<td>Design the architecture of the solution</td>
<td>Rational Software Architect</td>
</tr>
<tr>
<td>Web Services Developer</td>
<td>Implement the solution</td>
<td>IBM® Rational® Application Developer and IBM® WebSphere® Integration Developer</td>
</tr>
</tbody>
</table>

This series of articles focuses on how to capture business requirements, build services models that fulfill them, and create and deploy solutions that realize the designs. We will also highlight the supporting tools. They are intended to represent the minimum set of SOA modeling capabilities, currently offered by the IBM tools listed in Table 1, that can be used to model consumer requisitions and provider services in any architectural layer. These articles do not focus much at all on methods or techniques for discovering requirements, approaches for analyzing and evaluating services solutions, or approaches to partitioning services into various architectural layers. For further information on those important topics, see IBM® Rational Unified Process® (RUP®) for SOA and RUP for SOMA (see Related topics for links). These IBM® Rational® Method Composer plug-ins provide development processes, guidance, tool mentors, and articles that explain additional ways to use the tools to develop complete service models and solutions.

Purchase Order Process example

This example is based on the Purchase Order Process example taken from the OMG UML Profile and Metamodel for Services (UPMS) RFP, and it is based on an example in the BPEL 1.1 specification (see Related topics). The BPEL 1.1 spec contains only a partial solution, because correlation sets aren't defined, the business data is incomplete, and there is no fault handling or compensation. This version of the example includes some made-up data for completeness -- in particular, data needed for correlation.

The example starts by using IBM® Rational® RequisitePro® to describe the business motivation that establishes the business goals and objectives that are to be achieved. This is followed by a high-level business process captured using IBM® WebSphere® Business Modeler that expresses the business organizational and operational requirements necessary to meet the goals and objectives. These motivations and process models establish a context for identifying services that are connected to the business requirements.

After we understand the business requirements, we can proceed to the development of services that meets these requirements. The first step in an SOA solution is to identify the services. To do this, we will treat the business process as a Service Requirements contract. Then service specifications and service providers are designed and connected together in a manner that fulfills the business requirements while addressing software architectural concerns.

This process of identifying candidate services from a business model is also known as domain decomposition. IBM® Rational Unified Process® (RUP®) SOMA describes domain
decomposition and several other approaches which, when used collectively, provide heightened assurance of identifying all the services that are needed by the business.

**Business requirements**

**Scenario:** A consortium of companies has decided to collaborate to produce a reusable service for processing purchase orders. The goals of this project are to:

- Establish a common means of processing purchase orders
- Ensure that orders are processed in a timely manner and deliver the required goods
- Help minimize stock on hand and inventory maintenance costs
- Minimize production and shipping costs

Figure 1 shows the requirements captured in RequisitePro. Notice that the Process Purchase Order business use case meets all of our business goals.

**Figure 1. Business requirements shown in RequisitePro**

We also create a key performance indicator (KPI) for Goal 2: Timely order processing (see Figure 2).
Figure 2. Key performance indicator

This KPI will be something that we will want to observe in the business process model that realizes the business use case, will become a constraint in our SOA solution, and will be something that is monitored in our deployed Web service. This will allow the service to be monitored and managed to ensure that the business goals and objectives are actually met.

Business organizational processes

The business analysts from the member companies analyzed the requirements and determined that the following IBM® WebSphere® Business Modeler business process meets the business objectives, as well as business operational constraints. This process is intended to be sufficiently complete and detailed that it could be used as the basis of a formal service level agreement (SLA) between the parties. Therefore, our SOA solution that meets these business requirements should adhere strictly to these business requirements (Figure 3).

Figure 3. Purchase Order Process business process model

The Purchase Order Process initiates three parallel activities: one for managing production and shipping scheduling, another for price calculation and invoicing, and a third for shipping the ordered products. Processing starts by initiating a price calculation based on the products
ordered. This price is not yet complete, however, because the total invoice depends on where the products are produced and the amount of the shipping cost. At the same time, the order is sent to production to determine when the products will be available and from what locations. In parallel, the process requests shipping and then waits for a shipping schedule to be sent from a production scheduling provider. After the production schedule is available, the invoice can be completed and returned to the customer.

We used WebSphere Business Modeler to create a business measure called **Order processing time limit** for meeting the Order Processing Time Limit KPI from the business requirements. This business measure monitors the Purchase Order Process processing time and raises an alert if the processing time is greater than five days (Figure 4).

**Figure 4. Business measures for KPIs**

The business analysts may use WebSphere Business Modeler to simulate the business process. This would serve a number of important purposes:

- First, it could be used to enact the business process to envision how it would work. This would enable the business analysts to validate operations with the various stakeholders. Customers often do not know what they want until you show them something that isn’t it. Running a business process in simulation mode is a great way to get early validation of a business process before investing in a solution to the wrong problem. Envisioning the business processes through simulation may also help you discover new opportunities for refinement that are difficult to find by simply reading the process.

- The second thing that simulation can be used for is to determine how much executing a process will cost and how long it will take -- key issues that need to be considered when making business decisions. The business analyst can assign resources and durations to each of the tasks in the process. These resources can include availability, costs, and location information. WebSphere Business Modeler simulations can then be used to estimate how long a process will take, what it will cost, and where there might be resource bottlenecks that need to be addressed. Different simulation runs resulting from changes in the business process can easily be compared to determine whether the change has the desired effect.
Finally, the results of the simulation runs can be used to get early evaluations of the business KPIs to ensure the business processes meet the goals and objectives. This gives the development team the confidence they are building the right solution to the right problem at the right time.

We also linked the WebSphere Business Modeler Purchase Order Process to the BUC1 Purchase Order Process business use case in RequisitePro to indicate the process realizes the use case. "Realization" is a formal term in use case modeling. You can see that the business use case is linked to a process by the small arrow in the BUC icon in WebSphere Business Modeler’s Requirement Explorer view (Figure 5).

**Figure 5. Linking business processes to the business use cases that they implement**

![Diagram showing linking of processes and use cases](image)

You can use the **Requirements** perspective in WebSphere Business Modeler to navigate between the business process and the business use case it realizes in the **Requirement Explorer** view in WebSphere Business Modeler, the RequisitePro client, or the requirement in Microsoft® Word®.

The next section treats the business process as a Requirements contract and shows how to identify services in an SOA solution for the intended business operational requirements.

**Service identification**

Some business processes can be run directly on platforms, such as IBM® WebSphere® Process Server. But there are situations where business processes have not sufficiently addressed IT concerns and could possibly be refactored for better reuse and to address nonfunctional requirements, such as performance, scalability, and security. In this case, the business processes may be thought of as specifications of the requirements for what an SOA solution must do.

**Service requirements**

The requirements for the Purchase Order Process can be obtained from the WebSphere Business Modeler business process. These requirements are viewed as a service contract, indicating the roles that participate in the business process, the tasks that they are expected to perform, and the rules for how they interact. This service contract is a formal, architecturally neutral specification of the business requirements that are performed by interacting service consumers and providers,
without addressing any IT architectural or implementation concerns. In this case, the service contract contains the same information as the original business process; it is simply a view of that business process created by opening the WebSphere Business Modeler model using IBM® Rational® Software Modeler or IBM® Rational® Software Architect.

Business analysts usually use WebSphere Business Modeler; whereas, IT architects use Rational Software Architect. It is unlikely that these tools will be used together, and they typically are not used to access the same user workspace. To enable the IT architect to benefit from the business analyst's work, Rational Software Architect can be used to import WebSphere Business Modeler business models into an Rational Software Architect workspace, where he architect can then open it and see the business analyst's content rendered into equivalent UML. Figure 6 shows selections of our WebSphere model rendered in Rational Software Architect as UML.

Figure 6. Service Requirements contract

Let's spend some time understanding this diagram, because we'll be seeing similar diagrams as we develop the SOA solution.
The <<serviceCollaboration>> Process Purchase Order collaboration corresponds to the Process Purchase Order business process.

**Note:**
The collaboration is shown using the classifier notation (a partitioned rectangle), rather than the dashed ellipse notation. UML 2 allows either. This example uses the rectangle notation because it has more room for the roles.

When modeling service requirements or deriving service requirements from business processes, a collaboration answers these questions:

- **What effect is the requirement intended to accomplish?** This is the collaboration name that corresponds with the business process if the collaboration is derived from a business process. These names are often verb phrases that indicate what the collaborating roles are intended to accomplish.

- **Who participates to get it done?** The collaboration roles indicate who participates in the collaboration. These roles may correspond to swim lanes in a business process.

- **What are the roles responsible for?** (Note: Service providers will play these roles, not necessarily people.) The types of the collaboration roles generally are interfaces. The responsibilities of the roles are stated as the operations of the interface. Anything in our services solution that plays a role must be capable of performing these responsibilities. That is, they must implement those operations. In a business process, the roles are responsible for the tasks that are in their swim lanes. These interfaces could be derived by creating lists of tasks assigned to roles in the business process.

- **What roles interact?** That is, which roles have to communicate with other roles? This establishes dependencies between the roles. This interaction is shown by connectors between the roles in the collaboration. These connectors indicate that there is some control or data that flows across swim lanes in the business process.

- **What are the rules for how the roles interact?** This is the behavior owned by the collaboration. The behavior could be an activity, interaction, state machine, or opaque behavior (code, for instance). This behavior says when the responsibilities of the roles are requested and may indicate the information exchanged between them. The activity corresponds to a UML view of the business process.

- **How do we evaluate whether the requirements were met?** A collaboration may have constraints that specify the conditions that must be true for the requirements to be satisfied. These constraints correspond to the KPI business measures in the business process.

The Purchase Order Process is a service collaboration with four roles, including one played by the process itself. These roles are derived from the business process by examining the tasks assigned to the roles in the business process swim lanes. WebSphere Business Modeler uses a process-centered view of business requirements, whereas the services contract takes a role-centered view. This provides a more services-oriented view of business contracts, which makes it easier to bridge the gap between the process requirements and the architecture of service-oriented solutions.
The service contract may have constraints derived from the metrics and measures from the business motivation model. These constraints indicate what the service contract is intended to accomplish and how to measure success.

The service collaboration may also realize use cases that capture additional information about the high-level functional and nonfunctional requirements for the business process from the perspectives of the key external stakeholders or actors. These use cases can be viewed in use case diagrams in Rational Software Architect and linked to business use cases in RequisitePro. This maintains the link between the service requirements contract, business processes, the business use case, and the business goals and objectives.

The service collaboration contract shown in Figure 6 could have been a view of a WebSphere Business Modeler business process model, or it could have been developed directly in Rational Software Architect. Which approach to use is a matter of preference, of who is doing the modeling, and whether simulation is used to validate business processes. In either case, full traceability is maintained. Of course, the quality of the service collaboration contracts from WebSphere Business Modeler business processes will depend on the level of detail in those processes. A more complete discussion of the relationship between business process modeling and service modeling is provided in the developerWorks article, Business services modeling.

**Services project organization**

We are now ready to model a service to meet these requirements. Our solution will fulfill these requirements, but it is not necessarily constrained by them. When we design the SOA solution, we need to identify the services, design their interfaces, and decide how they will be implemented: Which service providers provide what services and how. This establishes the dependencies between the service consumers and providers and establishes the coupling in the system. Managing this coupling is a major part of designing SOA-based services. By separating the business requirements from the services, we have the flexibility to design the SOA to meet both the business and IT system requirements. This keeps the business requirements from overly constraining the SOA solution, which could lead to less reuse and less business agility.

First, we create a Rational Software Architect services modeling project called **PurchaseOrderProcessModel**. That project contains a services model called **PurchaseOrderProcess**. This model consists of an information model for the services, service data, specifications of the provided and required interfaces, and components providing the required services. But, for now, we are focused primarily on service identification.

As Figure 7 shows, there are five main packages in the PurchaseOrderProcess model:

- **org::ordermanagement** contains elements concerned with order management.
- **org::crm** contains the data model and common interfaces for some envisioned Customer Relationship Management standard that service consumers and service providers have agreed upon for developing shared services.
- **com::acme::credit** contains elements concerned with invoicing and credit management.
• **com::acme::productions** contains elements concerned with productions and scheduling.

• **com::acme::shipping** contains elements concerned with shipping.

These packages divide the problem domain into different functional areas. This helps manage complexity, establishes required name spaces, facilitates reuse, and keeps separate concerns in different packages (see Figure 7).

**Figure 7. Package dependencies diagram**

![Package dependencies diagram](image)

This organization could be called the dominant organization of the service model. Partitions may be used to document other classifications for organizing the same model elements.

**Services topology**

We start the service model by identifying the services that are involved in processing a purchase order. At this point, we are not looking for any detail about what the services do nor how they interact. We just want to create a sketch of what the services are and a high-level idea of how they might interact.

Figure 8 is a simple class diagram (or free-form diagram) in Rational Software Architect that shows the packages that represent the business functional areas described previously and the key services that will be defined in those packages. The only information given about the service at this point is its name.

The use dependencies in the diagram are intended to show how these services are intended to relate to each other. At this point, we don't have complete service specifications, nor do we know what service participants will provide or require what services. We also haven't modeled any implementations of these services. Therefore, there is no formalism for what these use dependencies mean yet. They are simply an indication of anticipated implementations that may or may not be true when we complete the service specifications and implementations. However, these dependencies are valuable at this high level, because they start to identify uses and coupling between systems that needs to be managed carefully.
**Figure 8. Service topology diagram**

![Service topology diagram](image)

**Note:**
How this service topology was discovered and designed is beyond the scope of this article. IBM RUP for SOMA (see Related topics) provides a process, methods, and guidance for how to discover services from business functional areas.

One simple way to discover the required services is to look at the business requirements identified as service collaborations and think about what service provider will be required to play the roles in the collaborations. That is the technique used for this simple example. We can actually formalize this by creating a diagram that shows how the service specifications would be used to fulfill the Service Collaboration contract.

Figure 9 shows an abstract component that models the context for processing purchase orders. It has several parts with types that are the service specifications discussed previously. The diagram shows a potential use of each of those service specifications in a particular context. We still don't know what service participant provides or uses these services, only that some participant does. These parts are also abstract, because they are instances of service specifications, rather than instances of service providers that provide these services.
**Figure 9. Meeting the Service Requirements contract**

What this diagram does show, however, is how these services fulfill the Service Collaboration requirements contract. The contract is a collaboration use, an instance of the Process Purchase Order service collaboration. The parts of the Processing Purchase Orders component are bound to the roles they play in that contract. This provides a formal link back to the business requirements and shows the business relevance of each of the services interfaces that we are intending to specify and implement. The behavior of the Process Purchase Order service collaboration specifies how these service parts have to interact in the service implementation. We will be revisiting this when we specify and implement the services to ensure that the solution meets the contract requirements.

**What's next**

In this article, we outlined a technique to identify services that are connected to business requirements. We started by capturing the business goals and objectives necessary to realize the business mission. We then modeled the business operations and processes that are necessary to meet the goals and objectives. Then we viewed the business process as a service collaboration that represents a Business Services Requirements contract that must be fulfilled by our eventual solution. We then used this requirements contract to help identify the required services and the potential relationships between them. This provides a formal way of identifying business-relevant services that are linked to the business goals and objectives that they are intended to fulfill.

The next article in this series, "Modeling SOA: Part 2. Service specification," will model the service specification of each of these services in detail. Those specifications will indicate the provided and required interfaces, the roles that those interfaces play in the service specification, and the rules or protocol for how those roles interact in providing the service.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSM sample model</td>
<td>RSM_sample_model.zip</td>
<td>60KB</td>
</tr>
</tbody>
</table>
Related topics

- **Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA** by Ali Arsanjani is about the IBM Global Business Services’ Service Oriented Modeling and Architecture (SOMA) method (IBM® developerWorks®, November 2004).
- **IBM Business service modeling**, a developerWorks article by Jim Amsden (December 2005), describes the relationship between business process modeling and service modeling to achieve the benefits of both.
- **UML 2.0 Profile for Software Services**, also by Simon Johnston (developerWorks, April 2005), describes the UML profile for software services, a profile for UML 2.0 that allows for the modeling of services, service-oriented architecture (SOA), and service-oriented solutions. The profile has been implemented in IBM Rational Software Architect.
- Download **IBM product evaluation versions** and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Subscribe to the developerWorks Rational zone newsletter. Keep up with developerWorks Rational content. Every other week, you'll receive updates on the latest technical resources and best practices for the Rational Software Delivery Platform.
- Download the trial version of **IBM Rational Software Architect V7**.

© Copyright IBM Corporation 2007
**Trademarks**
(www.ibm.com/developerworks/ibm/trademarks/)