Modeling with SoaML, the Service-Oriented Architecture Modeling Language: Part 1. Service Identification

Skill Level: Advanced

Jim Amsden (jamsden@us.ibm.com)
Senior Technical Staff Member
IBM

07 Jan 2010

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 OMG SoaML standard 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.

How modeling improves SOA

The power of a service-oriented architecture (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. SoaML models raise the level of abstraction to allow you to focus on business services. Model-driven development approaches can be used to generate the designed solution implementations for platforms such as Java™ Platform, Enterprise Edition (JEE), IBM® CICS®, or Web-Oriented Architectures (WOA) by using RESTful Web services 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 by 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 business and architectural decisions. To some extent, the approach that we will be describing applies one of the fundamental principles of SOA to business and solution development: separation of concerns and loose coupling. Here, we cleanly separate the tasks and responsibilities of business analysts from those of IT staff members.

Creating agile, timely, reusable IT solutions to business problems can be difficult. It requires a focus on architecture that separates concerns and reduces coupling between parts, especially when solution components are owned by different organizations. Being able to rapidly respond to change through innovative integrated business solutions is even harder. Integration and interoperability require standards for components developed by different organizations at different times to be plugged together into new solutions. **SoaML** (Service-Oriented Architecture Modeling Language) is an Object Management Group (OMG) standard that is intended to fill this gap and to help realize the potential of SOA. SoaML is a small set of extensions to UML to support SOA modeling. It provides an abstraction of SOA that focuses on describing the needs and capabilities of participants and connecting them in service value chains.

SoaML offers several benefits

- Enables interoperability and integration at the model level
- Provides a higher level of abstraction separate from platform variability

and the complexity of lower-level Web services XML document standards

- Addresses business integration and service interaction concerns at the architectural level by using architecture as the bridge between business requirements and automated IT solutions
- Enables SOA both on and between existing platforms through model-driven architecture (MDA)
- Allows for flexible platform choices
- Decouples solution architecture platform implementations to prevent existing solutions from inhibiting platform evolution
- Leverages and integrates with existing OMG standards for end-to-end life cycle development and management

About this series on modeling SOA

This series of articles shows how to use UML models that are extended with the OMG SoaML 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 SoaML details but will, instead, focus on how you can use SoaML to help with design and development.

Although this series of articles is about Web services solutions from SoaML 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 solution has to meet. We then use this business process to help identify needed business capabilities that can be realized as services in a designed 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® 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. In addition, we extend the unified modeling language (UML) for services modeling by adding the OMG SoaML Profile to the UML models in IBM® Rational® Software Architect. 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

Role	Task	Tools
Business executive	Convey business goals and objectives	IBM® Rational® Requirements Composer
Business analyst	Analyze business requirements	IBM Rational Requirements Composer
Software architect	Design the architecture of the solution	IBM® Rational® Software Architect
Web services developer	Implement the solution	IBM® Rational® Application Developer

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 demonstrating the services modeling capabilities currently offered by the IBM tools listed in Table 1. 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 SOMA (see Resources 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 SoaML standard, and it is based on an example in the BPEL 1.1 specification (see Resources). 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 Requirements Composer 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 also captured using Rational Requirements Composer that sketches 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) for SOMA describes domain decomposition and several other approaches which, when used collectively, provide heightened assurance of identifying all the capabilities and services that are needed by the business.

Business requirements

Usually, business analysts will focus 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 do not often have the capabilities necessary to address these concerns, and, if they did, they probably would not be effective tools for business analysts. In this section, we use sketches of business process models to identify the business capabilities needed to meet some business objectives.

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

- 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 Rational Requirements Composer. Notice that the Process Purchase Order business use case meets all of our business goals.

Figure 1. Business requirements shown in Rational Requirements Composer

Larger view of Figure 1.

We also create specific objective (or key performance indicator) that quantifies the "Timely order processing" goal (see Figure 2).

Figure 2. Objectives quantify goals

Larger view of Figure 2.

This objective 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 BPMN (Business Process Modeling Notation) business process captured in Rational Requirements Composer 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 contract 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

Larger view of Figure 3.

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 also linked the Purchase Order Process to the Purchase Order Process business use case to indicate the process realizes the use case (Figure 4). *Realization* is a formal term in use case modeling that indicates the relationship between a specification of something and its various implementations.

Figure 4. Linking business processes to the business use cases that they implement

You can use this link to navigate between the business process and the business use case that it realizes.

The next section treats the business process as a specification of business requirements and shows how to identify capabilities and services needed fulfill the business operational requirements.

Services project organization

We are now ready to create a model to meet these requirements. Our solution will

fulfill these requirements by identifying the capabilities, exposing appropriate capabilities as services, and then defining the architecture for how the services interact. When we design the solution, we need to specify the services, design their interfaces, and decide how they will be implemented (which service providers provide what services and how). This establishes the dependencies among the service participants 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 **PurchaseOrderProcess**. 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 5 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 that are 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 5. Package dependencies diagram

This organization could be called the *dominant* organization of the service model. Perspective packages can be used to document other ways of organizing the same model elements, such as by the SOMA method, for example.

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 IT solution must do.

Capabilities identifying candidate services

We start the service model by identifying the capabilities that are involved in processing a purchase order. At this point, we are not looking for any detail about what the services do or how they interact. We just want to create a sketch of what the required capabilities are and a high-level idea of how they might interact in order

to identify any additional capabilities.

The details for how these capabilities were identified is beyond the scope of this article. IBM RUP for SOMA (see Resources) provides a process, methods, and guidance for how to identify capabilities from business motivation and strategy, business functional areas, business processes, and existing assets. IBM SOMA describes three techniques:

- **Goal-service modeling**, which identifies capabilities needed to realize business requirements such as strategies and goals
- **Domain decomposition**, which uses activities in business processes and other descriptions of business functions to identify needed capabilities
- **Existing asset analysis**, which mines capabilities from existing applications

By using goal-service modeling and domain decomposition techniques, we can identify the capabilities needed to process purchase orders. Figure 6, which follows, is a simple class diagram in Rational Software Architect that shows the identified capabilities. The only information given about the capability is its name. These capabilities were identified by examining the business process and organizing the tasks into operations of capabilities identified for each swim lane in the process.

The use dependencies in the diagram are shows how these capabilities are intended to relate to each other. At this point, we do not know what capabilities might be exposed as services, we don't have complete service specifications, nor do we know what service participants will provide or require what services. We also have not modeled any implementations of these services. Therefore, these use dependencies are simply an indication of anticipated relationships between the capabilities that might or might 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 6. Service capabilities

We got the capabilities for the Purchase Order Process from the Rational Requirements Composer business requirements and processes. Service identification consists of determining which capabilities should be exposed as services. The IBM SOMA method provides a Service Litmus Test that can be applied to capabilities to determine which ones should be exposed as services. We do not cover those details here, so see RUP for SOMA for further information.

The Service Litmus Test would examine each capability and apply various configurable metrics to determine which one should be exposed as services. Figure 7 shows the services that were identified that expose the capabilities. For example, the InvoicingService service interface will expose the Invoicing capability. This means that any participant that provides this service will provide the capability, and any participant that requests it will use the capability.

Figure 7. Services identified for processing purchase orders

Services architecture

SoaML provides several ways of identifying services. The requirements from the business process could be viewed as a service architecture, thus indicating the participants in the business process, the service contracts that specify how they interact, and the choreography of their services and requests.

A service architecture is a formal specification of the business requirements that are performed by interacting service participants, without addressing any IT architectural or implementation concerns. In this case, the service architecture contains the same information as the original business process and can be treated as a specification for how to realize that business process.

Figure 8 shows the service architecture for processing purchase orders in IBM® Rational® Software Modeler. The details for the ServiceContracts referenced in Figure 8 are not covered in this article, but they will be addressed in subsequent articles in this series. For now, these ServiceContracts simply indicate the agreed-upon interactions between participants that play the indicated roles in the service architecture.

Figure 8: Services architecture for processing purchase orders

Larger view of Figure 8.

It helps to spend some time understanding this diagram, because we'll be seeing similar diagrams as we develop the SoaML solution.

The «`ServicesArchitecture`» Manufacturer Architecture corresponds to the Process Purchase Order business process.

Note:
The service architecture is shown using the classifier notation (a partitioned rectangle), rather than the usual dashed ellipse notation for a UML collaboration. 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 service architecture answers these questions:

- **What effect is the requirement intended to accomplish?** This is the service architecture name that corresponds to the business requirements. These names are often verb phrases that indicate what the collaborating roles are intended to accomplish.
- **Who participates to get it done?** The service architecture roles specify the **participants** who interact to achieve the desired results.
- **What are the roles responsible for?** The types of the roles represent service participants. The responsibilities of the participants are specified by the service contracts that connect them. Anything in our services solution that plays a role must be capable of performing these responsibilities. That is, they must implement operations specified by the service contracts
- **What roles interact?** That is, which roles have to communicate with other roles? This establishes dependencies between the roles. This interaction is shown by references to service contracts defining the permissible interactions between the participants.
- **What are the rules for how the roles interact?** This is the behavior owned by the service architecture. The behavior could be an activity, interaction, state machine, or opaque behavior (code, for instance). This behavior says when the services of the roles are requested and may indicate the information exchanged between them.
- **How do we evaluate whether the requirements were met?** A service architecture can have constraints that specify the conditions that must be true for the requirements to be satisfied. These constraints correspond to the business objectives.

The Purchase Order Process is a service architecture with four roles, including one played by the process itself. These roles could be derived from the business process by examining the tasks assigned to the roles in the business process swim lanes. Rational Requirements Composer uses a Business Process Modeling Notation (BPMN) process-centered view of business requirements; whereas, the service architecture takes a role-centered view. This provides a more services-oriented view of business requirements, which makes it easier to bridge the gap between the process requirements and the architecture of service-oriented solutions. Alternatively, the service architecture could have been designed to realize the capabilities that were identified from the business requirements and processes. In fact, a service architecture could actually be a different view of a BPMN business process.

The service architecture can have constraints derived from the business objectives.

These constraints indicate what the service architecture is intended to accomplish and how to measure success.

The service architecture can also realize use cases that capture 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 Requirements Composer or Rational Software Architect. This maintains the link between the service requirements contract, business processes, the business use case, and the business goals and objectives.

Whether you use business capabilities or service architectures, or both, to identify candidate services is a matter of personal preference. Business capability modeling is a straightforward way to decompose a business into competencies identifying the business capabilities and operations, and the relationships between capabilities needed to meet business objectives. Services architecture modeling provides a more formal way of specifying the participants and contracts governing their interactions. Both approaches can be used together. Use the approach most comfortable for you.

What's next

In this article, we outlined a technique to identify capabilities that are needed to meet business requirements. We started by capturing the business goals and objectives for the business vision. We then modeled the business operations and processes that are necessary to meet the goals and objectives. Then we used the business process to identify the required capabilities. This provides a formal way of identifying business-relevant capabilities that are linked to the business goals and objectives that they are intended to fulfill.

The next article in this series, "Part 2. Service specification," examines the capabilities and determines which ones should be exposed as services. We will then elaborate the service interfaces in more detail. Those interfaces will indicate the provided and required interfaces, the roles that the consumer and provider participants play in the service interface, and the rules or protocol for how those using the service operations.

Resources

Learn

- SoaML, an OMG standard profile that extends UML 2 for modeling services, service-oriented architecture (SOA), and service-oriented solutions. The profile has been implemented in IBM Rational Software Architect.
- Daniels, John, and Cheesman, John. *UML Components: A Simple Process for Specifying Component-based Software*. Addison-Wesley Professional (2000).
- 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.
- Using model-driven development and pattern-based engineering to design SOA: Part 2. Patterns-based engineering, Part 2 of a four-part IBM developerWorks tutorial series by Lee Ackerman and Bertrand Portier (2007).
- Design SOA services with Rational Software Architect, a four-part IBM developerWorks tutorial series by Lee Ackerman and Bertrand Portier (2006-2007).
- Model service-oriented architecture with Rational Software Architect: Part 3. External system modeling, Part 3 of a five-part IBM developerWorks tutorial series by Gregory Hodgkinson and Bertrand Portier (2007).
- Modeling service-oriented solutions is Simon Johnston's great article describing the approach to service modeling that drove the development of the IBM UML Profile for Software Services, the RUP for SOA plug-in (developerWorks, July 2005) and SoaML.
- SOA programming model for implementing Web services: Part 1. Introduction to the IBM SOA programming model, by Donald Ferguson and Marcia Stockton (developerWorks, June 2005), describes the IBM programming model for Service-Oriented Architecture (SOA), which enables non-programmers to create and reuse IT assets. The model includes component types, wiring, templates, application adapters, uniform data representation, and an Enterprise Service Bus (ESB). This is the first in a series of articles about the IBM SOA programming model and what is required to select, develop, deploy, and recommend programming model elements.
- Read SOA programming model for implementing Web services: Part 1. Introduction to the IBM SOA programming model, by Donald Ferguson and

Marcia Stock, to learn more about Service Data Objects, which simplify and unify the way applications access and manipulate data from heterogeneous data sources (developerWorks, June 2005).

- See Web Servoces for Business Process Execution Language for more about the BPEL 1.1 specification.
- 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.
- Browse the technology bookstore for books on these and other technical topics.

Get products and technologies

- Download trial versions of other IBM Rational software.
- Download IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Tivoli®, and WebSphere®.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.

About the author

Jim Amsden
Jim Amsden, a senior technical staff member with IBM, has more than 20 years of experience in designing and developing applications and tools for the software development industry. He holds a master’s degree in computer science from Boston University. His interests include enterprise architecture, contract-based development, agent programming, business-driven development, Java Enterprise Edition, UML, and service-oriented architecture. He is a co-author of =Enterprise Java Programming with IBM WebSphere (IBM Press, 2003) and of the OMG SoaML standard. His current focus is on finding ways to integrate tools to better support agile development processes. Jim is currently responsible for developing IBM Rational software’s Collaborative Architecture Management strategy and tool support.

Trademarks

Trademarked terms commonly used in developerWorks content are attributed on the

Trademarks page.