Integrating BPMN and SoaML: Part 3. Mapping BPMN and SoaML

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

18 November 2014

BPMN and SoaML are two recent standards adopted by OMG. There is significant overlap between these standards, with each providing a particular emphasis on service and process modeling. This article looks at how to uses their unique features together.

View more content in this series

Introduction

The first article in this series, Part 1: Integrating BPMN and SoaML: Motivation and approach, covered the motivation for integrating BPMN and SoaML. Integrating the standards allows modelers to use their complimentary capabilities for modeling behavior and structure to address a broader range of problems and support additional stakeholder views. Part 2: Integrating BPMN and SoaML: Concepts that guide integration, introduced the concepts used to guide the integration, how each language approaches encapsulation, contracts (or interfaces), structure and behavior.

This last article in this series, Part 3: Mapping BPMN and SoaML, uses the concepts from Part 2 as a context in which to do the commonality/variability analysis between the standards that informs the actual integration. The result of this analysis is captured in a simple mapping table that describes a set of relationship between concepts in SoaML and concepts in BPMN. This mapping can be used for a variety of purposes, including linking related, complimentary concepts across the standards. A concrete manifestation of this mapping as supported in IBM Rational Software Architect demonstrates links between BPMN process models and SoaML services models, and how these links address the requirements introduced in Part 1.

The mapping between BPMN and SoaML, and the realization of that mapping in Rational Software Architect allows you to integrate BPMN and SoaML in a manner that leverages their complimentary capabilities and views to drive greater value than can be achieved from using either of them individually.

Review of Part 2: Concepts that guide integration

BPMN supports process models, including orchestration of activities in a business process, collaboration between processes, and the choreography of the interactions between processes. SoaML supports services architectures and the encapsulation of interactions between participants in a services architecture using service contracts to model service-level agreements (SLA).
Part 2 of this series explored how SoaML and BPMN support concepts such as encapsulation, contracts, structure and behavior in order to understand the similarities and differences between the languages, and inform potential mappings between them.

The concepts covered were:

- **Encapsulation**: A description of the encapsulating element or component including its potential interactions with other prototypical components. SoaML uses the Participant component while BPMN uses the Participant pool.
- **Contract:** The encapsulation and specification of the potential interactions or interfaces between components and the agreements the components are expected to adhere to, including message exchange patterns, grouping and choreography. SoaML uses a ServiceContract and ServiceInterface to define interactions between participants while BPMN uses orchestration, collaboration, conversation, choreography and simple service interfaces.
- **Structure:** The Internal structure of an encapsulating component including the assembly of other components. SoaML uses a Participant's internal structure to specify the service channels connections between parts that represent references to instances of other participants. BPMN does not distinguish between type and instance, and uses collaboration, conversation and choreography to also describe structure.
- **Behavior:** A Representation of a component's internal behavioral implementation or orchestration, including how the component uses and/or provides services according to the agreed upon contract. SoaML uses any UML behavior including activities, state machines, interactions or opaque behaviors while BPMN defines behavior using processes.

Mapping BPMN and SoaML

Table 1 summarizes the relationship between BPMN and SoaML. This information can be used to:

- Better understand the concepts of BPMN or SoaML by examining them from different viewpoints
- Inform model-to-model transformations used to translate BPMN to/from SoaML in order to exchange models between supporting tools
- Define relationship links between model elements captured in one language with related concepts in another. For example, a SoaML participant owned behavior could be linked to a BPMN process that describes the implementation of that behavior possibly using OSLC links (see open-services.net).
- Provide guidance to process and service modelers who want to make optimal use of both modeling languages.

The intent of this mapping is to highlight the key integration points between BPMN and SoaML, not to define an exhaustive mapping that could be used to transform or interchange models from one

language to/from the other with no loss of information or semantic meaning. The intent is to focus the mapping on possible linkage between specifications, to show their complimentary capabilities rather than their overlap.

Many other mappings can be defined using different conventions for what BPMN lanes and pools represent, different approaches to modeling message exchange patterns, a different scope and/or a different level of detail. The design of this mapping was driven by the requirements addressed in part 1 of this series, and the concepts covered in part 2. Other mappings might be required to address different requirements, purposes or needs.

Table 1. The relationship between BPMN and SoaML

SoaML concept	BPMN concept
ServicesArchitecture	Overview choreography
Participant	Participant representing PartnerEntity (within definitional collaboration, denoted by a pool)
Service port	A lane in a pool that represents one end of a conversation between participants in a communication diagram: Interface of the above participant (the line connecting the hexagon and the pool)
Request port	A lane in a pool that represents the other end of the conversation, the one that sends the first message
ServiceInterface	Interface, (no service protocol required). Alternatively, a conversation in a conversation diagram, including the corresponding messages in a collaboration diagram, and the choreography of those messages in a choreography diagram
Interface (realized or used by a ServiceInterface)	Interface used to define operations for service tasks
Operation or Reception (of an Interface)	Operation of an Interface or Message
Parameter (of an Operation)	Message inputs and outputs for an Operation
Activity (or Behavior)	Process
CallOperationAction (including target InputPin and parameter Pins)	ServiceTask
SendSignalAction	SendTask
AcceptCallAction	InitialNode
AcceptEvenAction	InitialNode
ControlFlow	Sequence Flow
ObjectFlow	Data Association

Existing tool support

This section of the article explores existing tool support for integration between SoaML and BPMN that utilizes some of the concepts in the mapping table from the previous section. The intent is to provide a concrete example of integration between the notations and models, and to demonstrate how this integration provides additional value.

IBM® Rational® Software Architect provides support for UML 2, SoaML and BPMN. See the five-part series of articles Design and develop a more effective SOA written by Gary Johnston and Todd Dunnavant. These articles describe the supported integration between BPMN and SoaML including:

- Link from BPMN data object to UML type
- BPMN ServiceTask can refer to an operation of a UML Interface
- Traceability link from a SoaML Capability to BPMN Process describing how the capability is provided
- Link between UML OpaqueBehavior and BPMN Process as a means of using BPMN to describe a UML Behavior
- OSLC links between UML model elements and BPMN processes through Rational Design Manager to support links to other lifecycle artifacts including requirements (including BPMN process sketches, use cases and screen flows), work items included in project iteration plans, and test cases

These integration capabilities in Rational Software Architect provide support for the BPMN and SoaML integration requirements listed in part 1 of this series, and as described in the following sections of this article.

Requirement 1: Identify SoaML Capabilities (or candidate services) from BPMN processes
With Rational Software Architect users can create a traceability link from a SoaML Capability to a BPMN process to indicate the BPMN process implements the capability. Rational Software Architect Design Manager also supports creating OSLC links between SoaML capabilities and BPMN processes in Rational Software Architect and other IBM process modeling tools. See Requirement 2 for details.

- It is also possible to create a SoaML Capability directly from a BPMN process. To do this, create a SoaML class or free form diagram
- Select the palette entry **Services** > **Capability** from BPMN element.
- Navigate to and select a process in a BPMN model (a .bpmx diagram)

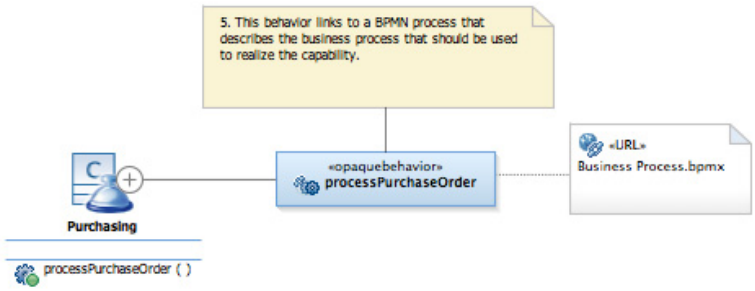
A new SoaML capability is created with operations that correspond to all the activities in the BPMN process. Edit and remove any operations that are not applicable to this capability. A URL link is created from the SoaML capability to the BPMN process that implements the capability. Select the capability, right-click and select **Navigate** > **To URL** to open the properties of the linked element. From there you can easily navigate to the linked BPMN process in the project explorer. Similar steps can be used to create SoaML Capabilities from BPMN lanes and tasks.

Requirement 2: Identify, specify and implement services using SoaML.

Rational Software Architect provides complete support for SoaML and limited support for the IBM SOMA method. A SoaML Capability represents a capability provided by a business organization that may expose services the organization provides in order to deliver a value proposition meeting client needs. A Capability can own behaviors that describe how the capability should be provided. You can use a UML OpaqueBehavior to describe how a capability should be provided using simple text. The opaque behavior can also be linked to a BPMN process that models how the capability should be implemented.

The link can be implemented using a URL link from the UML opaque behavior to a BPMN process in the modeler's workspace, or using OSLC links through Rational Software Architect Design Manager, as shown in Figure 1.

Figure 1. Linking a Capability to a BPMN Process



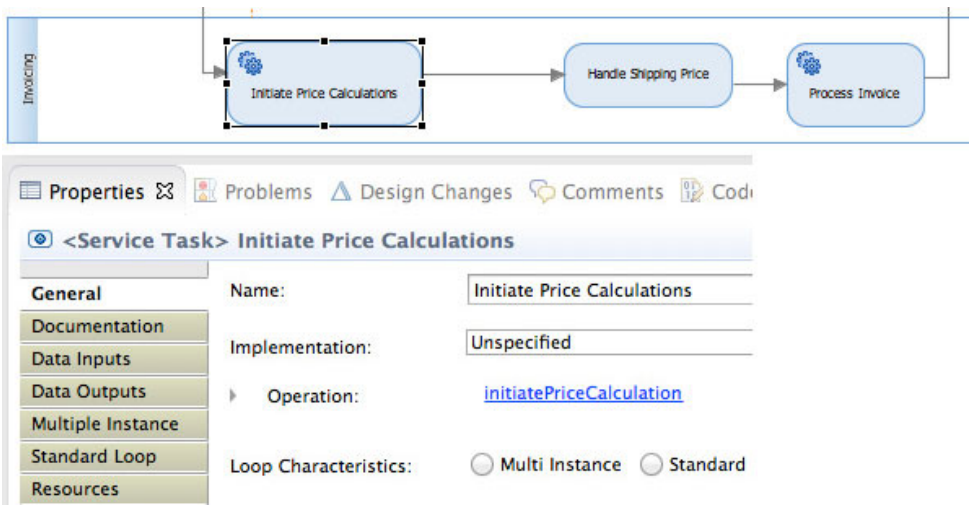
These links allow BPMN processes to create or link to SoaML Capabilities which represent candidate services. The Capability can then be linked to Participants and ServiceInterfaces that implement and expose the capability to potential clients. As a result, SoaML can be used to fully implement services that have been identified from BPMN processes.

Requirement 3: A BPMN Service Task can invoke a SoaML service operation.

The Rational Software Architect BPMN editor allows the Operation of a BPMN Service Task to link to a UML operation in a SoaML service interface. In this case, the lane should identify the interface containing the operation.

In the example shown in Figure 2, the Initiate Price Calculations service task in the Invoicing lane is linked to the InitiatePriceCalculation of the SoaML Invoicing service interface.

Figure 2. Initiate Price Calculations service task



Requirement 4: Implement a SoaML service operation provided by a Participant using a BPMN process.

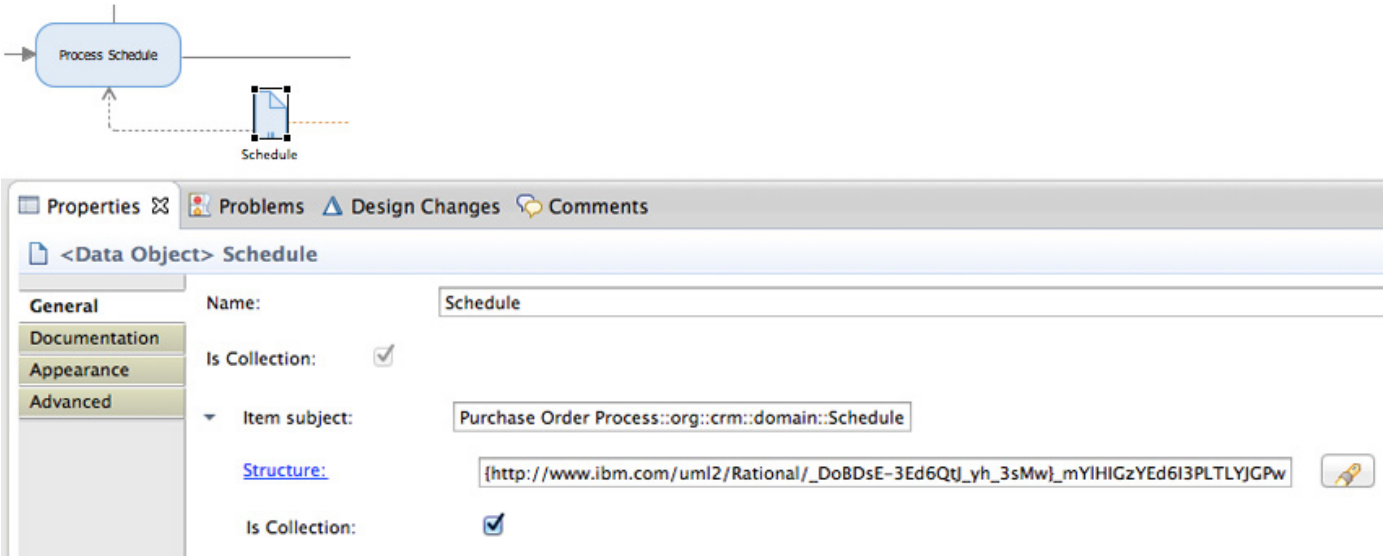
Similar to linking SoaML capabilities to BPMN processes, Rational Software Architect supports the creation of a URL link between a UML opaque behavior and a BPMN process. This can be used to indicate that the behavior is implemented using BPMN. Rational Software Architect Design Manager also supports creating OSLC links between a Participant and a BPMN process, and the BPMN process to a SoaML service operation. This could be used to indicate the BPMN process is a method implementing the operation provided by the participant.

In this case, the BPMN process consists of a pool representing the SoaML Participant, with lanes representing the service and request ports. These linkages are not directly supported, but could be if consistent naming conventions indicating the relationships between the different model elements is used.

Requirement 5: Share information specifications (classes, data types, messages) between BPMN and SoaML.

Rational Software Architect provides the ability to define the structure of data item definitions which in turn model the content of BPMN messages, data objects or data stores. To create the link, select a **message**, **data object** or **data store** on a BPMN diagram, then select the **General** tab in the Properties view. Expand the **Item subject property**, and click the **search button** (the icon that looks like a flashlight) to the right of the Structure property. Navigate to the UML class that defines the item subject. A URL link is created between the item definition and the data store as shown in Figure 4.

Figure 3. BPMN data store described by a UML class

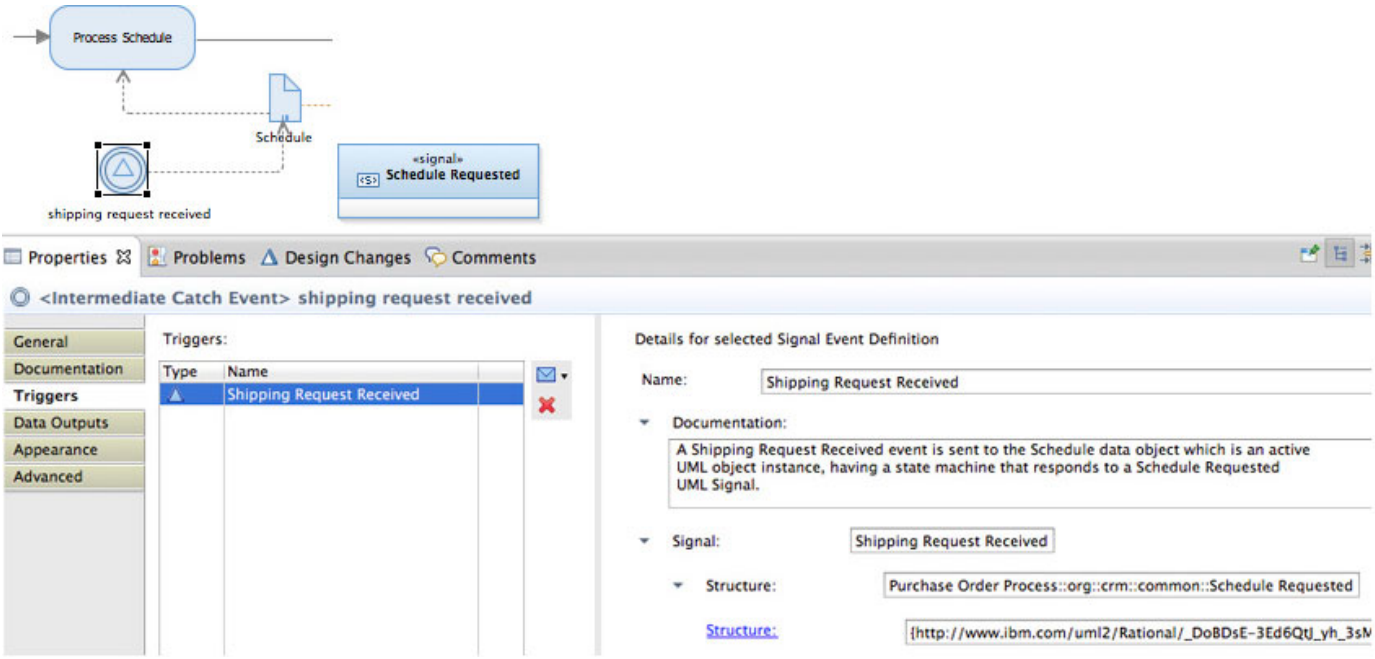


Requirement 6: Use SoaML to define lifecycles for data objects in BPMN that can respond to business events

A BPMN intermediate catch event triggered by a signal can be connected to a data object that represents a UML class with an active lifecycle. The Structure property for the signal can be set to an item definition that links to the corresponding UML Signal. That signal can then be used in a state machine that is an owned behavior of the active class to indicate how instances change their

state in response to the event. Figure 5 shows the shipping request received signal is connected to the Schedule Data Store and defined by the Schedule Requested signal.

Figure 4. BPMN signal sent to data store



Conclusion

BPMN and SoaML are two important standards adopted by OMG. There is significant overlap between these standards, with each providing a particular emphasis. BPMN focuses on process models, starting with simple orchestration of activities in a business process, and expanding to support collaboration between processes, and then expanding again to support choreography of the interactions between processes. SoaML focuses on the encapsulation of interactions between participants in a services architecture using service contracts to model service-level agreements (SLA). This series of developerWorks articles explores how to use these two modeling languages, either separately or together, on the same or related projects, and how to do so in a way that leverages their unique, complimentary features while avoiding redundancy resulting from their overlap.

Resources

Learn

- Modeling SOA: Part 1. Service identification by Jim Amsden is the first in a series of five articles about developing software based on service-oriented architecture (SOA). (IBM developerWorks, October 2007).
- Modeling SOA: Part 2. Service specification by Jim Amsden is the second in a series of five articles about developing software based on service-oriented architecture (SOA). (IBM developerWorks, October 2007).
- Modeling SOA: Part 3. Service realization This third article of this five-part series explains how SOA-based Web services are actually implemented. (IBM developerWorks, October 2007).
- Modeling SOA: Part 4. Service composition This fourth article of this five-part series covers how to assemble and connect the service providers modeled in "Part 3. Service realization" and choreograph their interactions to provide a complete solution to the business requirements. It also shows how this service participant fulfills the original business requirements. (IBM developerWorks, October 2007).
- Modeling SOA: Part 5. Service implementation In this final article of a five-part series, you will learn how to create an actual implementation that is consistent with the architectural and design decisions captured in the services model. Learn to generate the platform-specific implementation by exploiting both model-driven development and the IBM Rational Software Architect UML-to-SOA transformation feature to create a Web service from the SOA. (IBM developerWorks, October 2007).
- 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).
- Design and develop a more effective SOA, Part 1: Introducing IBM's integrated capabilities for designing and building a better SOA This is the first article in a five-part series on IBM's commercial solution for service-oriented systems design and development. This article begins by discussing some of the promises and issues associated with moving to a service-oriented approach for IT systems. High-level descriptions of best practices and tools for realizing the benefits and overcoming the issues are also explored. (IBM developerWorks, May 2011)
- IBM Business service modeling This article describes the relationship between business process modeling and service modeling to achieve the benefits of both. (IBM developerWorks, December 2005)
- OMG BPMN Specification and OMG SoaML Specification. Learn more about these specification by downloading them in PDF format.
- Browse the technology bookstore for books on these and other technical topics.

Get products and technologies

- Download the trial version of 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.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.
- Rational Software Architect, Data Architect, Software Modeler, Application Developer and Web Developer forum: Ask questions about Rational Software Architect.

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.