IBM

developerWorks®

Integrating BPMN and SoaML: Part 1. Motivation and approach

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

22 July 2014

BPMN and SoaML are two recent standards adopted by OMG. The standards overlap significantly, but each provides a particular emphasis on service and process modeling. This article describes how to use their unique features together.

View more content in this series

In this three-part series on BPMN and SoaML integration, learn how to use the strengths of each standard to complement the other. The articles in this series explain how to:

- Specify the requirements for BPMN and SoaML integration that takes advantage of their complementary capabilities.
- Better understand the concepts of BPMN or SoaML by examining them from different viewpoints.
- Inform model-to-model transformations that can be used to translate BPMN to and from SoaML to exchange models between supporting tools.
- Define relationships between model elements captured in one language with related concepts in another. For example, an SoaML participant-owned behavior can be linked to a BPMN process that describes the implementation of that behavior.
- Make optimal use of both modeling languages.
- Enable more effective modeling practices to achieve better value from the models, to understand complexity, to improve planning, and enjoy other benefits.
- Facilitate the specification of more refined reusable assets.
- Promote the existing integration capabilities, supported by OSLC specifications, in vendor tools.
- Provide a foundation for extending the current OSLC specification for additional linking capabilities.

Motivation for integrating BPMN and SoaML

The Object Management Group (OMG) has approved Business Process Modeling Notation (BPMN) 2.0, a significant update to the BPMN 1.0 notation to support process orchestration, collaboration, and choreography. OMG has also approved SOA Modeling Language (SoaML) 1.0, a profile extension of Unified Modeling Language (UML) 2 for modeling Service-Oriented Architectures (SOA). These standards overlap in some areas, but each provides a particular emphasis.

BPMN focuses on process models. It starts with simple orchestration of activities in a business process, expands to support collaboration between processes, and expands again to support choreography of the interactions between processes.

SoaML focuses on services architectures and the encapsulation of interactions between participants in a services architecture by using service contracts to model service-level agreements (SLA). Because SoaML is an extension of UML, services models can include behavior in services architectures, service contracts, and service participants to model the interactions between service consumers and providers. The models also cover how the services are implemented and used.

Although these standards cover many of the same topics, they deal with structure and behavior from different perspectives. Each standard has unique features in its semantics and notations, which are often reflected in the supporting tools and platforms. The challenge is how to use these two modeling languages (either separately or together, on the same or related projects), how to take advantage of the unique, complementary features, and how to avoid redundancy.

This article is the first in a three-part series that describes ways to integrate BPMN and SoaML and maximize the advantages of both. Explore the purpose of integrating them, learn the requirements for the integration, and discover an approach for analyzing the similarities and differences between the modeling languages.

Part 2 describes how each language addresses:

- **Encapsulation**: A description of the encapsulating element or component, including its potential interactions with other prototypical components
- **Contract:** The encapsulation and specification of the potential interactions between components and the agreements the components are expected to adhere to
- **Structure:** The internal structure of a encapsulating component, including the assembly of other components
- **Behavior:** A representation of a component's internal behavioral implementation or orchestration, including how the component consumes and provides services according to the agreed-upon contract

This analysis provides the basis for the proposed integration that is covered in Part 3. Modelers can use this information to determine which language to use for what purposes and how they can be used together effectively. Tool vendors can use the information to provide links between supporting tools to support the integration, to provide additional modeling options, and to yield better traceability and impact analysis. OMG members can use the information in these articles to inform requests for proposals (RFPs) for future development of the standards.

Structure and behavior

Structure and behavior are a means of distinguishing the complementary capabilities of SoaML and BPMN. Consider external and internal views of a domain or problem space. An *external view* describes an entity's outward facing or specification view. This view includes both demand- and supply-side components that specify goals and value propositions, needs and capabilities, and expectations and commitments. An *internal view* describes a design of the information and activities an entity has chosen to accomplish its goals. It includes the behaviors to deliver its value propositions consistent with its commitments.

Together, these views address the organization, information, and behaviors from different perspectives: structural specification and behavioral implementation. To understand the implication of these different views, focus on the stakeholders that are primarily concerned with them. The integration of BPMN and SoaML must support better collaboration between these roles.

Roles of people who use BPMN and SoaML

Business analysts can use BPMN to analyze business processes to identify new, adapted, or updated capabilities and services. They can also create business processes that choreograph or orchestrate these capabilities and services after they are implemented. This task is sometimes known as service consumption.

Business analysts generally are not concerned with, nor capable of defining, services architectures that adequately address IT solution concerns such as distribution, persistence, availability, transaction scopes and integrity, security, and similar concerns. BPMN does provide sufficient modeling capabilities to support this perspective.

Solution architects are responsible for developing an effective, efficient solution architecture that meets current and anticipated requirements. They might use UML, one of the primary OMG specifications for Model Driven Architecture (MDA), extended with the SoaML profile as a general-purpose modeling language. SoaML supports a broad range of capabilities for business and solution analysis, including design and construction of capabilities and services. Integrating BPMN and SoaML gives development teams a complete solution for service construction and consumption.

This integration is especially useful in enterprise architecture management, because it provides a means of connecting business architecture building blocks with the information systems building blocks that realize them. Both sets of building blocks employ SOA style to minimize coupling between building blocks in the enterprise. Integrating these modeling languages provides the context for exploring detailed requirements with BPMN and SoaML.

Requirements of the BPMN and SoaML integration

The integration must meet the following requirements.

Requirement 1: Business analysts can sketch BPMN processes, either using simple orchestration, or using more complete collaborations, conversations, or choreography with other process participants. These processes can be treated similar to UML use cases for requirements analysis. Each task in a swim lane, each lane in the process, or each process can represent a requirement for some business activity. These tasks can then be used to identify candidate services that are to be used to realize the tasks. Business analysts must be able to create realization relationships between a BPMN task, lane, or process and one or more SoaML ServicesArchitecture, Capability, or ServiceInterface elements.

Requirement 2: After the SoaML capabilities have been identified, SoaML can be used to specify service interfaces that expose the capabilities. Participants providing the services can be developed that have methods that implement them.

Requirement 3: Services that are defined by SoaML ServicesInterfaces and provided by participants in SoaML can then be invoked using ServiceTasks in a BPMN process. This ability makes it possible for BPMN to take advantage of reusable services defined by SoaML.

Requirement 4: The method (or implementation) of a service operation owned by a participant can be modeled as a UML Behavior in SoaML. A Behavior can be an Activity, Interaction, StateMachine, or OpaqueBehavior. Behaviors owned by the participant are methods of the service operations provided by the participant, and use service operations required by the participant. It must be possible to use a BPMN process to describe the method that implements a service operation.

Requirement 5: BPMN can define messages and data objects. SoaML can define classes, data types, primitive types, and message types. SoaML must be able to be used as a means of defining information for BPMN, and any information defined in a BPMN process must be able to be used in SoaML service interfaces and methods.

Requirement 6: Some business entities, for example, cases in case management systems, can have lifecycles that are often modeled by state machines. SoaML must be able to be used to define lifecycle models for these business entities, including the events they respond to, the resulting actions, and how their state changes over time. These active business objects must be able to be used in BPMN processes, and process activities must be able to send events to the objects and understand the state they are in.

In summary, BPMN and SoaML integration must support the construction and consumption of services through the separation of the architecture of services from the processes that define, implement, or use them. With this integration modelers can:

- Identify SoaML Capabilities (or candidate services) from BPMN processes
- Identify, specify, and implement services using SoaML
- Use BPMN to define a method to implement an Operation of a SoaML ServiceInterface provided by a Participant
- Invoke a service operation, defined by a SoaML ServiceInterface and provided by a SoaML Participant, as a ServiceTask in a BPMN process
- Share information specifications (classes, data types, messages) between BPMN and SoaML
- Use SoaML to define lifecycles for data objects in BPMN that can respond to business events

Approaches to integration

One approach to defining complementary views is to examine structure and behavior. SoaML provides modeling capabilities that are useful for defining the structure of systems of collaborating systems. BPMN has better features for defining dynamic behavior. SOA is a way of understanding business and technology in terms of loosely coupled entities that provide and use each other's capabilities through services. It is way of understanding the business and the technology. In terms of high-level representations of the business at the "C" level, the processes used to deliver the organization's products and services are often not the primary driver. Rather, the products and services the processes deliver are the focus, followed by collaborations within the supply chain.

From another viewpoint, business analysts often sketch business processes needed to realize specific business tactics that support the achievement of stated business goals. They use the activities in these processes to help identify other participants in the supply chain and their roles. In negotiation, activities can be moved from an internal orchestration to different partners and suppliers to determine the best operational strategy. A task might move from something a participant does to a request of some other participant. The required services and service participants can be incrementally identified from these business processes.

Businesses don't exist if they don't provide something of value to their customers, and they can't provide anything without processes that get the work done. Therefore, BPMN and SoaML can be seen as two parts (structure and behavior) of the same coin, rather than competing ways of performing the same task. SoaML provides the outward facing view of an entity's responsibilities and what it requires of others through encapsulation of the structure of the participants and the interactions between them. BPMN provides a design of the activities an entity has chosen to do to accomplish its goals, including providing and using services. Behavior- or structure-centered concepts and views are both useful at the business and technology level.

Depending on the situation and stakeholders involved, there might be a preference for one view or the other, just as there is sometimes a preference to start with data instead of processes or services. In the end however, all these views have to fit together: structure, processes, information, events, and services. The challenge is to find ways to use the OMG standards, associated methods, and tools in a manner that makes it possible for modelers to use the best notations and semantics for modeling different aspects of their problem domain and solutions. You can use the complementary capabilities of these standards to your advantage rather than choosing one over the other.

Guiding principles to choose an integration approach

The following guiding principles can help you develop an approach to integration:

- Identify any issues in the BPMN or SoaML specifications:
    - Improve the quality of the specifications.
    - Improve your understanding of the specifications.
    - Facilitate integration, regardless of how it is implemented.
- Minimize undesirable coupling between the specifications and their supporting meta-models.

- Ensure that the specifications continue to stand alone and be used independently, as well as together.
- Minimize implementation costs for vendors and users.
- Focus on complementary modeling capabilities, rather than similarities and overlaps.
- Identify concepts and topics that fulfill requirements without regard to BPMN or SoaML specifics:
  - What is the structure and behavior of the collaborating service consumers and providers?
  - What services are provided and who are the consumers of those services?
  - How are the services are implemented and used?
- Use specific examples to explore similarities and differences.
- Take advantage of the strengths and avoid the weaknesses of each specification.
- Favor semantically rich links between elements in the standards rather than information transformation and exchange between them.
- Ensure support for information exchange.

Integration must avoid introducing undesirable coupling between the specifications to ensure that the individual capabilities can be used independently or together as needed.

Because BPMN and SoaML were developed independently by different teams (with some overlap) and because both are now adopted specifications, the integration has not been designed into each of the standards as they were developed. Although the OMG specification lifecycle process does encourage integration between specifications, submission teams are often motivated by specific RFP requirements, limited resources, and tight schedules. These factors tend to focus specification development activities on the specific RFP requirements, sometimes at the expense of integration with other standards.

As a result, BPMN and SoaML integration must be addressed after the specifications have been completed and approved. It is possible but often impractical to accomplish effective integration of standard specifications through the revision process or through new RFPs. Tools vendors and users have invested in supporting products and models that can limit future flexibility. This situation leaves three remaining approaches to integration.

Integration approach 1: No integration

This approach assumes it is not necessary to integrate BPMN and SoaML. Each specification is already supported by multiple tools and can be used to solve problems. Although integration offers some potential value, each of the specifications can be used effectively to solve problems without this additional value. This approach assumes specifications and tools are independent with intentional overlap. This approach does not attempt to use the complementary capabilities of the specifications, but the overlap can be addressed somewhat by documenting best practices for using each specification.

Integration approach 2: Model interchange

In the typical model or information interchange, a transform converts a user's model from one form to another. This conversion makes it possible for the information to be created in one tool and used in another. Modelers can use the capabilities of both specifications and their supporting tools, but not together. This approach has minimal impact on the existing standards and tools and therefore can have a low barrier to entry. Consider the following implications of this approach:

- The mapping between the specifications must be fully informed by the meaning of each specification and an agreement about how that meaning is preserved through the transformation.
- The source specification can support content and semantics that are not covered by the target. In this case the extra content is lost, or it has to be managed by an informal approach using structured comments, annotations, or other extension mechanisms.
- Model-to-model transformations result in separate copies of the same model information. This information redundancy can significantly increase lifecycle management and reconciliation issues when trying to keep multiple copies of the same information, in different formats, synchronized.
- Because different mappings can exist for different purposes, it can be difficult to standardize how the interchange is to be done. Such standardization is important, because it enables compatible tools to be reliably implemented by multiple vendors.
- The cost of developing, maintaining and using the transforms can be high for vendors and end users.

To minimize the impact of these issues you can adopt methodologies or best practices that reduce the need to edit derived artifacts exchanged between the tools. For example, BPMN can be used to specify a high-level computation independent model (CIM) of the business. Then a model-to-model transform can be used to create an initial platform independent model (PIM) in SoaML for use in elaborating the SOA models for subsequent platform specific model (PSM) and code generation. This approach uses different notations and tools for different roles to take advantage of the strengths of each for different aspects of the overall project lifecycle. The details of this approach are beyond the scope of this article, but model-to-model mappings can be quite useful.

The model interchange approach focuses on the overlap between specifications and tools and on different ways of performing the same tasks, rather than taking advantage of the complementary capabilities afforded by each specification. Further, it does not accommodate using these capabilities to be used together. A third approach is needed.

Integration approach 3: Metamodel integration

This approach provides the best integration, but at higher development costs. Rather than modify either the BPMN or SoaML specifications, a new, bridging metamodel can be created that extends and links the specifications. This bridging metamodel can be created using, for example, UML 2 package merge, a MOF metamodel, a UML profile, or Open-Services for Lifecycle Collaboration (OSLC) specifications. Then BPMN or SoaML supporting tools can be extended to support the bridging metamodel or a separate tool can be created that supports both specifications and the integration.

Integration can be through either model interchange between BPMN and SoaML, integration at the metamodel level, or through bridging metamodels. All of these approaches are feasible, but the model interchange tends to focus on the overlap between BPMN and SoaML, where they are different representations and metamodels of the same thing. Metamodel integration focuses more on how they can be used together and complement each other, and how they might provide a greater value proposition. Each modeling language can be used for what it does best and the information can be linked and shared between them. For example, metamodel integration makes it possible for vendors to create tools that use BPMN and SoaML together in a standard way. Model bridging is similar to metamodel integration but is done using a separate mediator and bridge metamodel that results in less coupling between BPMN and SoaML. This method makes it possible for them to be developed and used separately, too.

Conclusion

BPMN and SoaML are two recent standards adopted by OMG. Although there is significant overlap between these standards, each provides 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 services architectures and the use of encapsulation of interactions between participants in a services architecture using service contracts to model service-level agreements (SLAs).

This article is Part 1 of a three-part series that explores how to use these two modeling languages (either separately or together) on the same or related projects. It introduces requirements and approaches for how to take advantage of their strengths of each and how to avoid redundancy.

Resources

Learn

- "Modeling SOA: Part 1. Service identification" (Jim Amsden, developerWorks, October 2007): First in a series of five articles about developing software based on service-oriented architecture (SOA).
- Modeling SOA: Part 2. Service specification (Jim Amsden, developerWorks, October 2007): The second in a series of five articles about developing software based on service-oriented architecture (SOA).
- Modeling SOA: Part 3. Service realization (Jim Amsden, developerWorks, October 2007): The third article of this five-part series explains how SOA-based Web services are actually implemented. The service implementation starts with deciding what component will provide what services. After these decisions have been made, you can model how each service functional capability is implemented and how the required services are actually used. Then you can use the UML-to-SOA transformation feature included in IBM Rational Software Architect to create a Web service that can be used in IBM WebSphere Integration Developer to implement, test, and deploy the completed solution.
- Modeling SOA: Part 4. Service composition (Jim Amsden, developerWorks, October 2007) The 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.
- Modeling SOA: Part 5. Service implementation (Jim Amsden, developerWorks, October 2007) The fifth article in this series shows how to create an actual implementation that is consistent with the architectural and design decisions captured in the services model.
- Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA (Ali Arsanjani, developerWorks, November 2004) is about the IBM Global Business Services' Service Oriented Modeling and Architecture (SOMA) method.
- IBM Business service modeling (Jim Amsden, developerWorks, December 2005) This article describes the relationship between business process modeling and service modeling to achieve the benefits of both.
- "Design and develop a more effective SOA, Part 1: Introducing IBM's integrated capabilities for designing and building a better SOA" (developerWorks, May 2011): 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. It then provides high-level descriptions of best practices and tools for realizing the benefits and overcoming the issues.
- OMB 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.

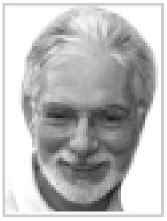
- Download IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

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 is an IBM Senior Technical Staff Member and Solution Architect. Jim works on a broad range of solutions incorporating IBM products for municipal strategic planning; solution analysis, design and construction; operational efficiency and monitoring; and business intelligence - tying these together into a comprehensive offering supporting the IBM Smarter Cities initiative. Jim has spent many years developing standards and products for modeling to understand and manage complex systems. He holds a Masters degree in Computer Science from Boston University and a Bachelors degree in Mathematics from the University of Maine. His interests include semantic web technologies and information integration and sharing, Enterprise Architecture, contract based development, agent programming, business-driven development, JEE, UML, and service-oriented architectures. He is co-author of Enterprise Java Programming with IBM WebSphere and an author of the OMG SoaML Specification, a standard for modeling Service Oriented Architectures. Jim has also done extensive research and development in the area of mathematical models for power generation optimization for paper mills. He is a TOGAF-Certified Enterprise Architect.

© Copyright IBM Corporation 2014
(www.ibm.com/legal/copytrade.shtml)
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)