UML 2.0 Profile for Software Services

By Simon Johnston, Architect, IBM Corporation.

Abstract

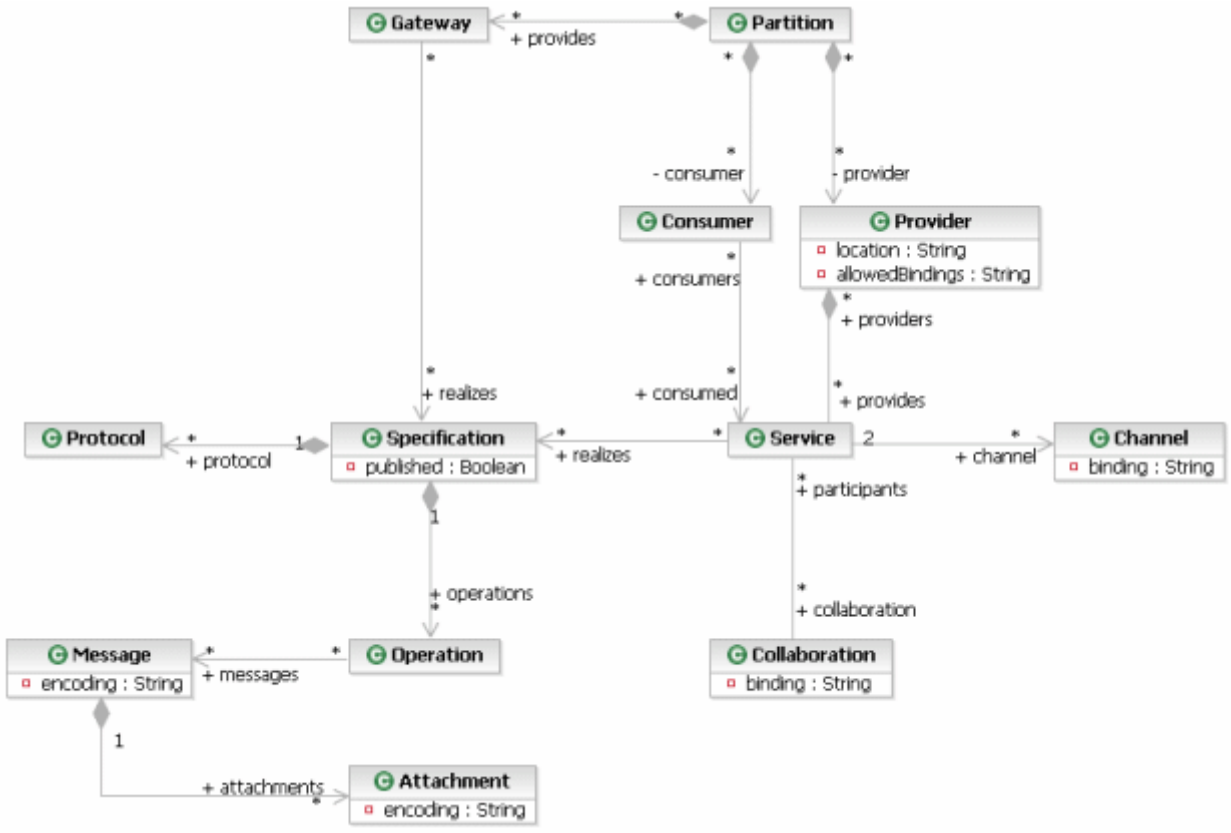
This paper describes the UML Profile for Software Services, a profile for UML 2.0 which allows for the modeling of services, SOA and service-oriented solutions. The aim of the profile is to provide a common language for describing services that covers a number of activities through the development lifecycle and also provides views to different stakeholders. So, for example, the profile provides capabilities for the architect to map out services early in the lifecycle using logical partitions to describe the entire enterprise-wide service portfolio. This view is further detailed by designers who develop the service specifications, both structural and behavioral that act as the contracts between the service clients and implementers. The message view provides the ability for designers to reuse information models for common service data definitions.

The profile has been implemented in Rational Software Architect and used successfully in developing models of complex customer scenarios and also in educating people to the concerns relevant to development of service-oriented solutions.

Overview of the UML Profile for Software Services

Conceptual Model

The following diagram is a model showing the concepts important in modeling services. As you can see the number of concepts is relatively small and should be reasonably familiar to anyone having worked on service-oriented solutions. Note, however that although the profile is a realization of this model a number of the concepts are not explicit stereotypes in the profile. For example there is no stereotype for operation or for protocol as these are existing notions in the UML 2.0 that the profile reuses without any ambiguity or further constraint.



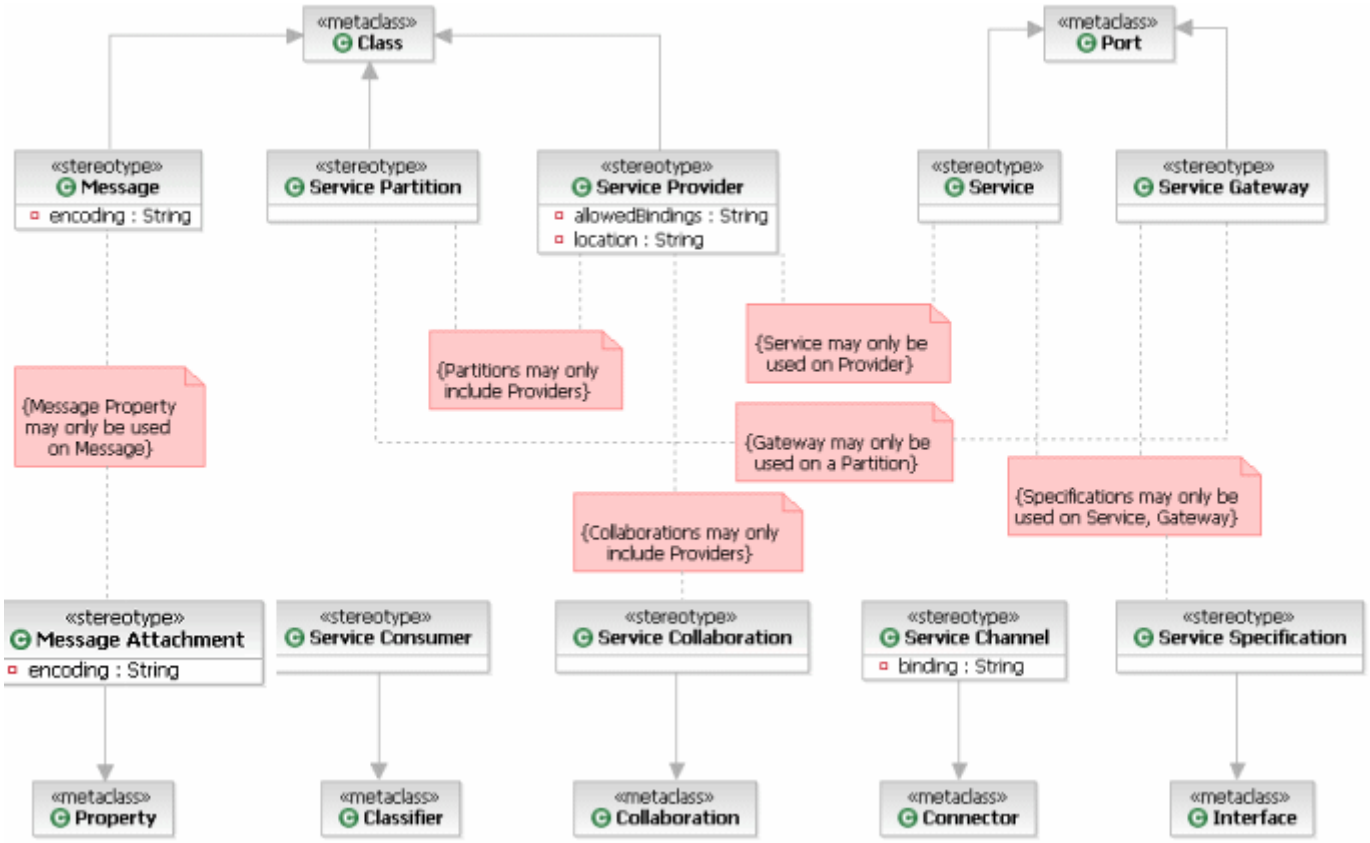
Identified Subset of UML 2.0

The following table lists the elements of the UML 2.0 meta model that are used as meta classes for stereotypes in the UML Profile.

UML 2.0 Meta class	Stereotypes
Class	Message, Service Partition, Service Provider
Classifier	Service Consumer
Collaboration	Service Collaboration
Connector	Service Channel
Interface	Service Specification
Port	Service, Service Gateway
Property	Message Attachment

The Profile Itself

The following (clickable) diagram is a UML 2.0 profile diagram, it demonstrates the actual details of the profile with each stereotype, its meta class using the extension notation (filled arrow head).You can also see some of the constraints in the model, particularly those co-constraints between profile elements.



The following sections outline the elements of the UML 2.0 profile as it is currently defined. Each section outlines an individual stereotype; the details of each specify its meta class, properties and any constraints which should be applied when using the profile.

stereotype Message

Extends

Class

Semantics

A message represents the concept as defined in the WSDL specification, i.e. a container for actual data which has meaning to the service and the consumer. A message may not have operations, it may have properties and associations to other classes (one assumes classes of some domain model). A message stereotype has a property to denote its assumed encoding form (i.e. `SOAP-literal`, `SOAP-rpc`, `ASN.1`, etc.).

The use of this element may be optional in a tool for two reasons. Firstly the modeler may simply wish to use elements from a domain model directly as the parameters to an operation rather than specifying a message. Secondly the modeler may wish to use the convention of specifying a set of input and output messages on an operation, in which case the modeling tool would have to construct an input and output message matching the parameters when generating service descriptions in WSDL.

Properties

Kind	Name	Type	Description
Property	encoding	String	Denotes the platform encoding mechanism to use in generating the schema for the message; examples might be `SOAP-RPC`, `Doc-Literal`, `ASN.1`, and so on.

Notation

Constraints

- Shall not have any owned operations.
- Shall not have any owned behaviors.
- All owned properties shall be public.

stereotype Message Attachment

Extends

Property

Semantics

This stereotype is used to denote that *some component of a messages is an attachment to it* (as opposed to a direct part of the message itself). In general this is not likely to be used greatly in higher level design activities, but for many processes attached data is important to differentiate from embedded message data. For example, a catalog service may return general product details as a part of the structured message but images as attachments to the message; this also allows us to denote that the encoding of the images is binary (as opposed to the textual encoding of the main message).

Properties

Kind	Name	Type	Description
Property	encoding	String	Denotes the platform encoding mechanism to use in generating the schema for the message; examples might be `SOAP-RPC`, `Doc-Literal`, `ASN.1`, and so on.

Notation

Constraints

- Shall only be used on properties owned by classes stereotyped as <<Message>>..

stereotype Service

Extends

Port

Semantics

The service model element provides the end-point for service interaction (in web service terminology) whereas the definition of these interactions are a part of the service specification. In the model a service not only identifies the provided interface, but may also identify required interfaces (such as callback interfaces). A service has an additional property that denotes the binding to be used, such as `SOAP-HTTP`, `SOAP-JMS`, and so on.

Properties

None.

Notation

Constraints

- Shall only be used on a Class stereotyped as <<Service Provider>>.
- Shall be typed by an Interface stereotyped as <<Service Specification>>.

stereotype Service Channel

Extends

Connector

Semantics

A channel represents *the communication path between two services*. It is important to note that interaction may occur over a channel, but the channel does not represent any particular interaction. In the web services world, each service denotes the binding(s) associated with it (so that a client may access it). In a modeling profile, you denote binding either on the communication between services or between a service and consumers. In this way, you can be flexible in understanding the binding requirements .

Properties

Kind	Name	Type	Description
Property	binding	String	Denotes the platform binding mechanism to use in generating the service binding in WSDL; examples might be `SOAP-RPC`, `SOAP-Doc`, `HTTP-Get`, and so on.

Notation

Constraints

- At least one end of the connector shall connect to a port stereotyped as <<Service>>.

- At most one end of the connector may connect to a port stereotyped as <<Service Gateway>>, a class stereotyped as <<Service Provider>> or classifier stereotyped as <<Service Consumer>>.

stereotype Service Collaboration

Collaboration

Extends

Semantics

A service collaboration is a *way of specifying the implementation of a service as a collaboration of other services*. From a web services point of view this corresponds to the use of BPEL4WS in specifying service implementation. So, this means that a service collaboration is used as the behavior of a service and, if it is intended to generate to a language such as BPEL -- it may have other implementation-specific constraints.

Properties

Kind	Name	Type	Description
Property	Binding	String	Denotes the platform binding mechanism to use in generating the collaboration as a process choreography; examples might be "BPEL", "WSFL", etc.

Notation

Constraints

- Collaboration participants shall be either <<ServiceConsumer>>, <<ServiceProvider>> or <<ServiceSpecification>> elements.

stereotype Service Consumer

Classifier

Extends

Semantics

Any classifier (class, component, ...) *may act as the consumer of a service*, and that includes another service. While this stereotype is most definitely optional it may be useful in identifying elements of a model -- that are not services themselves -- as *clients of services*. On the other hand it may be overhead and not used.

Properties

None.

Notation

Constraints

None.

stereotype Service Gateway

Extends

Port

Semantics

A service gateway looks like a service but is *only available for use on partitions and not service providers*. A gateway acts as a proxy service and can be used to mediate protocols or denote the interface available to a partition. For example, we might denote that although a number of services are implemented within a partition -- only some are available for use outside the partition, and therefore gateways are provided for these services. This disallows other services or partitions from communicating to services that are not exposed via gateways.

Properties

None.

Notation

Constraints

- Shall only be used on a Class stereotyped as <<Service Partition>>.
- Shall be typed by an Interface stereotyped as <<Service Specification>>.

stereotype Service Partition

Extends

Class

Semantics

A partition represents *some logical or physical boundary of the system*. It is optional to model partitions but useful. For example partitions could be used to represent the web, business and data tiers of a traditional n-tier application. Partitions might also be used to denote more physical boundaries (such as my primary data center, secondary site, customer site, partners and so on), in which case the crossing of partitions may have particular constraints for security, allowed protocols, bandwidth and so on.

A partition may only have properties that represent nested parts, be they services or other partitions. Note that *this is a constraint* - no other elements may currently be represented in a partition.

A partition also has the notion of being "strict", if a partition denotes that all communication between it and other partitions must be directed through typed gateways then it is said to be a *strict partition*.

Properties

Kind	Name	Type	Description
Property	Classifier	String	A classification name, to denote the namespace scope of this partition.

Notation

Constraints

- Shall not have any owned properties.
- Shall not have any owned operations.
- Shall not have any owned behaviors.
- Any owned Part shall be either <<Service Consumer>>, <<Service Provider>> or <<Service Specification>> elements.

stereotype Service Provider

Extends

Class

Semantics

The Service Provider is a *software element that provides one or more services*. In modeling terms one would most usually expect to see a UML component here, however such a restriction seems arbitrary and so the metaclass is noted as Class for more flexibility. A service provider has a property that captures information about its location although the meaning of this is implementation dependent. The Class acting as the service provider may not expose any attributes or operations directly, only public ports may be provided (stereotyped as service) and these are typed by service specifications.

The location property, while implementation/platform specific is useful in generating service endpoint names. For example with WSDL the location may be `http://svc.myco.com/` and a service might be called `CustInfo`, in which case the endpoint name for the service could be generated as `http://svc.myco.com/CustInfo`.

Properties

Kind	Name	Type	Description
Property	allowedBindings	String	Denotes the allowed platform binding mechanism a channel may use in connecting to the service; examples might be `SOAP-RPC`, `SOAP-Doc`, `HTTP-Get`, and so on.
Property	location	String	The location of the provider, this may be used by generators to create endpoint names.

Notation

Constraints

- Shall not have any owned properties.
- Shall not have any owned operations.
- Shall not have any owned behaviors.
- Any owned Port shall be stereotyped <<Service>>.

stereotype Service Specification

Extends

Interface

Semantics

The use of an interface *denotes a set of operations provided by a service*. Note that a service may implement more than one interface. By convention it is possible to attach a protocol state machine or UML 2.0 Collaboration to such a specification to denote the order of invocation of operations on a service specification. With such a behavioral

specification any implementing service can be validated against not only a static but dynamic specification of its structure and behavior. Note that the service specification may only provide public operations.

Properties

Kind	Name	Type	Description
Property	published	Boolean	This property denotes whether the service is assumed to be published into a service repository; this is a different notion from the public/private property provided by UML.

Notation

Constraints

- Shall not have any owned properties.
- All operations shall be marked as "public".
- May only be used as the type for a Port stereotyped as <<Service>> or <<Service Gateway>>.