JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 8, November-December 2008

An MOF2-based Services Metamodel

Harshavardhan Jegadeesan, SAP Labs, India
Sundar Balasubramaniam, BITS, Pilani, India

Abstract

As Service-Oriented Computing is gaining mainstream adoption, Services are emerging as core-building blocks of today's applications. In particular, web services have become the most common technology manifestation of the service-oriented computing paradigm. Basic open-standards that enable web services such as WSDL, SOAP etc. have evolved and stabilized over a period of time. However issues such as service composition, policy definition and enforcement, support for semantics are among the few issues that still remain open. With increased adoption of service-oriented computing and rapidly evolving technologies and standards to address these open-issues, heterogeneity has emerged in service development approaches leading to complexity and risks. In this paper, we address this problem by introducing modeling abstractions that could be used in the early-stage service development lifecycle of web-based electronic services. We present a holistic approach to services modeling using six model views. These views represent different perspectives of services modeling and form our core Services Metamodel with a grounding in the formal foundations of MOF2.

1 INTRODUCTION

Service-Oriented Architecture (SOA) [1] considers services as first-class entities to build applications. Services are self-describing, self-contained components that can be automatically discovered and invoked using open-standards. The biggest business motivation for SOA lies in the fact that companies could increasingly focus on their core competencies and look for business partners to support them with other context functions to create value for customers – leading to business process outsourcing [2]. This has led to breaking down of existing vertical industry structures and ushered in the phenomenon of networked businesses. Companies can now open up their platforms to business partners and affiliates to create value-additions for customers. They can expose business functions as remotely accessible services in a services marketplace.

Current service-oriented computing efforts are pre-dominantly technology-driven.There are a number of issues that have to be addressed before realising the vision of a services marketplace. We summarize a few of those issues in the view of services development, provisioning and consumption.

Firstly, metadata associated with services is lean and incomplete. The WS-* standards [3] [4] [5] which describe various facets of service metadata are semantically weak. For example, to access an eBay® web service, a *developer key* and a *merchantID* (obtained while signing up with the eBay® developer program [6]) must be supplied. These have to be supplied in the SOAP Header [7] for each service invocation. This information is not a part of the formal service description but is specified in the developer documentation. Since not all service facets can be adequately described by existing formal service description mechanisms, automated ways of service usage is still not a reality. We call this the *Lean Service Metadata Problem*.

Secondly, there is a rapid evolution of standards and technology resulting in disparate service development approaches. Though basic web service technologies like WSDL [8] (for service description) and SOAP [7] (for service invocation) have evolved and stabilized, associated specifications (WS-*) are still evolving and are likely to result in more competing standards. Also, alternate provisioning approaches and architectures like REST [9] have created more heterogeneity in the services ecosystem. As the standards and underlying technologies evolve at a rapid pace, the longevity of the solutions built on them reduces. We call this the *Evolving Standards Problem*.

Thirdly, unlike the software development approaches, in the services world business process experts and domain experts (a.k.a. business experts) must be able to define and model services from a business perspective. It is easy for these 'business users' to work with visual models rather than with a multitude of formal XML [10] based specifications, as is the norm today.

Services Metamodel

Efforts to address the *lean service metadata problem* and *evolving standards problem* along with emerging alternate service provisioning approaches would lead to heterogenity in services development. In order to improve the longevity of the solution and rigorously represent all facets of services, we need to capture the solution space in a platform and technology independent way using conceptual models. These models must be rich enough to capture associated services metadata irrespective of the whether the current standards support them. These models must be easy for business users to visually model services in the early stages of services development.

We follow the prescription of Model-Driven Architecture (MDA) [11] to specify services precisely using technology-agnostic, high-level conceptual models. These models can later be translated to concrete executable specifications or code using standard-mappings.

Our contribution in the paper is the following: Firstly, we identify different perspectives of services modeling and present six model views to support modeling of services. Secondly, we define an MOF2-based [12] Services Metamodel (a M2-level model in the 4-layer UML hierarchy) to model these different perspectives of services modeling. Our Services Metamodel extends the *UML Infrastructure Library::Core* package [13] (hereinafter known as Core) (fig 1).

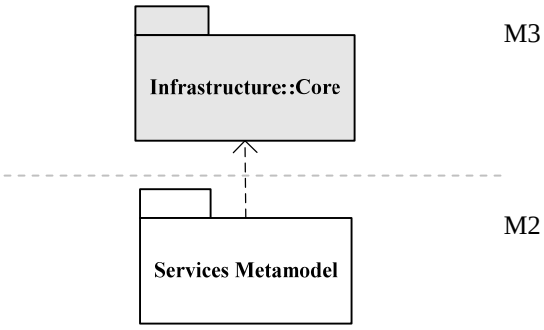


Fig 1. Services Metamodel

The rest of the paper is organized as follows: In section 2, we provide high-level requirements for services modeling; In section 3, we provide a motivating example justifying our high-level requirements as well as establish necessary perspectives for service modeling; In section 4, we provide the formal foundations of the services metamodel grounded in MOF2; In section 5, we use the services metamodel to model the example described in section 3; In section 6, we address conformance issues with reference to SOA-RM and W3C-Arch; In section 7, we analyze related work and finally provide conclusions and future work in section 8.

2 HIGH-LEVEL REQUIREMENTS

Services Modeling involves representing various facets of service requirements and solutions identified during early-stage services development. Modeling of services must be supported by a formal Services Metamodel. A Services Metamodel must not only enable capturing of different perspectives of services, but also support maximum expressiveness with a small set of modeling elements. Our focus - in this effort - is on the early-design phase of services development, especially web-based electronic services. With this in mind we identify the following high-level requirements for a services metamodel:

*R1: The metamodel shall enable capture of the high-level description of the service.

*R2: The metamodel shall enable capture of the different roles and perspectives of the actors associated with a service.

*R3: The metamodel shall enable capture of realization of services.

*R4: The metamodel shall enable capture of the operational details of a service in use.

*R5: The metamodel must be conformant to the Oasis SOA Reference Model (SOA-RM)

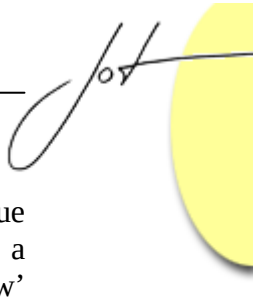
Requirements R1-R4 correspond to a subset of mandatory requirements in the OMG's RFP (Request for Proposal) – the UML Profile and Metamodel for Services [14]

(hereinafter referred to as RFP-UPMS). The high-level description of a service (R1) include ownership information, service capabilities and the roles involved in exercising these capabilities. Service Interfaces and the operations on service interfaces with their pre-conditons and post-conditions must also be a part of the service description. These correspond to RFP-UPMS mandatory requirement Service Description (requirement 6.5.7). The metamodel must support different roles of actors associated with the service (R2). Roles may include those of providers, consumers, aggregators and mediators. These requirements correspond to the RFP-UPMS mandatory requirements Service Provider and Service Consumer (requirements 6.5.12, 6.5.13). The metamodel must support realization mechanisms of services (R3). The realization mechanisms could include implementation by service providers or through aggregation of already existing services by an aggregator. These correspond to the RFP-UPMS mandatory requirements Realization, Composition, and Extension (6.5.14, 6.5.15, 6.5.17). The metamodel must support provisioning of services (R4) over many channels, deployment and invocation mechanisms for the service. These correspond to the RFP-UPMS mandatory requirement Invocation (6.5.9). In addition, our metamodel also meets the requirements on UML Compatibility (6.5.2) and Platform Independence (6.5.3).

3 MOTIVATING EXAMPLE

Our example is based on a real life scenario – eBay® Auctions [15] (however the services and the scenario described here are completely fictitious). eBay® allows auctioning of a variety of items based on certain rules and policies. Sellers can auction items by choosing a minimum bid amount and duration. Bidders bid for the item and the bidder with the highest bid at bid closing wins. The winning bidder pays the seller and the seller ships the item to the buyer. Both the buyers and sellers rate each other and the rating determines their credibility in the marketplace. eBay® supports business services such as Auctioning, Bidding and Rating but collaborates with business partners for Payment (Paypal®) and Shipping services (UPS®). There could be other partners in the services marketplace providing these services.

On the other hand, eBay® would also want to open-up its eCommerce platfom for businesses and affliates by exposing their business functions as services. This would enable a manufacturing company to auction its excess inventory through eBay® auctions directly from its Supplier Relationship Management (SRM) software like mySAP® SRM [16]. To expose a business function as a consumable service, a business expert must be able to specify the broad definition of this service in a technology agnostic fashion. Consider the ‘AuctionItem’ service which allows sellers to list an item in eBay® auctions. A business expert from eBay® needs a model view for defining the service, its broad purpose and the associated ownership domain. The ownership domain represents a logical partitioning of the services for administrative or deployment purposes. We need a *Service Definition View* to support the business expert in defining a service. The service definition view must support classification of the services as ‘atomic’, ‘composite’ or ‘abstract’. Atomic services represent atomic business functions such as ‘AuctionItem’.



Composite services aggregate other services and through this packaging improve value proposition to consumers. Consider the 'BuyItNow' service from eBay® which lets a seller directly sell the item at a fixed price instead of auctioning it. The 'BuyItNow' service in turn uses the 'ProcessPayment' service from Paypal® and 'ShipItem' service from UPS®. The 'BuyItNow' service which aggreagtes the 'ProcessPayment' and the 'ShipItem' services provided by business partners is a composite service. Lastly, a ownership domain must be able to define services that represent a business need – a gap in the value-chain – yet to be provided by any service provider. The reasons for defining an abstract service are the following:

1. An ownership domain would like its business partners to provide it with this service in its own terms and conditions (expresession of intent to outsource the service)
2. The ownership domain would want to defer the realization of the service

For example, assume that eBay® wants to introduce a new 'ValidateAuctionItem' service which would validate certain items being auctioned (e.g. art). Since eBay® might not have the expertise to do this they would want to outsource the realization of this service. The intention to outsource the service could be expressed by defining an abstract service.

The next step after service definition for the 'AuctionItem' service would be to define more concretely the capabilities provided by this service. How does the interface for auctioning an item look like? What are the different service properties? (e.g. cost of access, availability etc.). We need a *Service Capability View* which would define service properties and capabilities. The view must describe service interfaces, their service operations and the syntax associated with invoking these operations along with the schema of the messages and the message exchange pattern between the service provider and a consumer.

Once the basic capabilities and properties are defined, it must be possible to specify policies (such as 'security policy', 'auction policy' or 'service disruption policy'). The security policy could state that only registered and authorised users must be allowed to access the 'AuctionItem' service. The auction policy could state that perishable items could not be auctioned. We need a *Service Policy View* to define policies on services. Exact mechanisms to realize policies is specified by the IT team during realization by enhancing the policy definitions. For example, they could decide that the authorization (security policy) should happen through a signed certificate (e.g. X.509 digital certificate).

The 'AuctionItem' service once defined and capabilities expressed must be realized by the IT team using underlying IT assets (packaged applications, custom home-grown systems or mainframes.). Realizing a service could either be through an existing or new implementation – in case of atomic services (by service providers) or through service composition – in case of composite services (by service aggregators). We need a *Service Realization View* to capture this.

Every service consumer has a goal (a.k.a. need); a service offering should satisfy the goal of the service consumer. The relationship between consumer goals and service

offerings is an *n x m* relationship. Sometimes there may not be a direct 'fit' between the goals and services due to inherent heterogeneties resulting in the need for mediation. For example, the 'BidForItem' service which is used to bid for an auctioned item could be re-purposed to support a proxy-bidding scenario. In a proxy-bidding scenario, the system alters the bid for an item on behalf of the bidder based on user-specified rules. A service mediator could support this proxy-bidding scenatio. We need a *Service Mediation View* to support specification of mediation.

Once the abstract definition for a service is specified followed by the service realization, it must be possible to define external interaction points through which a service consumer could access the service. It must be possible to define various ways of binding to the service through the use of different transport protocols. It must also be possible to define service invocation properties. We need a *Service Deployment View* to describe the service interaction points and service invocation mechanisms.

From our example scenario, we have identified the six model views: service definition view, service capability view, service policy view, service realization view, service mediation view and the service deployment view. These six views represent different perspectives of services modeling.

4 SERVICES METAMODELS – THE SIX MODEL VIEWS

In this section we present the formal semantics of our Services Metamodel – the six model views and their corresponding modeling elements.

Service Definition View

The service definition view (fig. 2) defines a service, its purpose along with the ownership domain which owns the service. The ownership domain provides a logical partitioning of services in terms of physical or administrative boundaries. The business entity which owns the service could be the top-level ownership domain. Enterprise-wide service portfolio could be organized under hierarchies of ownership domains. Ideally, the business expert uses the service definition view as a starting point to define the ownership domain and the services they own.

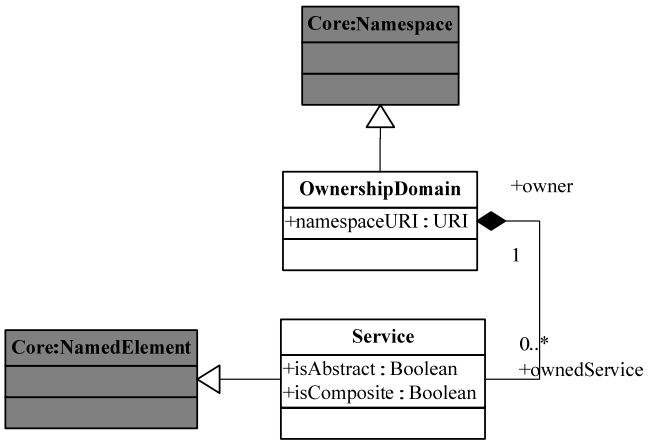


Fig 2. Service Definition View

Key Classes and Associations

Service: A service represents a capability of a provider which meets goals of consumers. It is a first-class modeling entity in our Services Metamodel. Service extends the *Core: NamedElement* from UML infrastructure Library (*Core* represents the UML Infrastructure Library). A service could be an atomic or a composite (isComposite = true) or an abstract service (isAbstract = true).

Ownership Domain: An ownership domain represents partitioning of services based on physical deployment or administrative domains. Ownership domain has owned services associated with it. Ownership Domain can in turn belong to another ownership domain thereby creating a hierarchy of ownership. The OwnershipDomain extends the *Core:Namespace*. The OwnershipDomain also has a namespace URI (uniform resource identifier).

Constraints

None

Service Capability View

The service capability view (fig. 3) helps in defining the capabilities and properties of a service which is defined using the service definition view. Using this view it is possible to define the service description, service properties, the service interface and the various service operations along with their pre- and post-conditions.

Key Classes and Associations

Service: Service has a property ‘isExtensible’ which determines if the service could be extended or enhanced. Extension of a service could either be functional enhancements (extending its capabilities) or property enhancements (enhancing service properties or policies associated with a service). Every service has an one or more service descriptions.

Service Description: A service description has a semantic description of the service and also includes a classification of the service. The classification system could be based on an existing system such as the North American Industry Classification System (NAICS) [17]. Service Description is associated with a Service Property and a Service Interface. A service could have multiple service descriptions, also a service description could exist without any service realizing it.

Service Property: A service property represents properties of service such as cost of access, availability and service rating. The property could be quantitative (describing a measure) or qualitative. One or more service properties are associated with every service description. Service properties are also used in identifying appropriate services during service discovery.

Service Interface: A service interface represents the underlying capabilities brought to bear by a service. A service interface could be extended to support specialization of a service. For a service to be extended, the 'isExtensible' property must be set 'true'. Every service interface has a set of supported operations and an exception associated with it. The ServiceInterface extends the *Core:Classifier*.

Service Operation: A service operation represents an underlying capability. Event-driven scenarios could also be modeled using a *notification* or an *event receiver* operation. A notification operation sends out messages that represent a notification, whereas an event receiver operation receives messages representing an event. Every operation has input and output messages and the sequencing of these messages is determined by the message exchange patterns (as defined in [18]). Marking an operation as 'isNotification' or 'isReceiver' could determine the message exchange pattern.

Service Constraint: A service constraint represents pre-conditions or post-conditions on service operations. A constraint could be a hard constraint (mandatory constraint) or just a preference (best-effort constraint). Service Constraint extends the *Core:Constraint.* A service constraint is owned by an OwnershipDomain.

Service Exception: A service exception represents an exceptional condition in a service operation execution. Every service operation would have an 'infault' or an 'outfault' message based on the message exchange pattern. A message exchange pattern defines the order of the messages between the provider and the consumer. An exception could also be defined at the level of a service interface. A service exception could be a 'resumable' exception – an exception does not halt the further execution of an operation after being handled properly.

Message: A message encapsulates input and output data for a service operation. We use the terminology 'message' as it is indicative of a loosely-coupled communication between service providers and consumers. The messages that are exchanged must be strictly typed and hence Message extends *Core: TypedElement.* The message label identifies whether a message is an input message or an output message.

Constraints

None

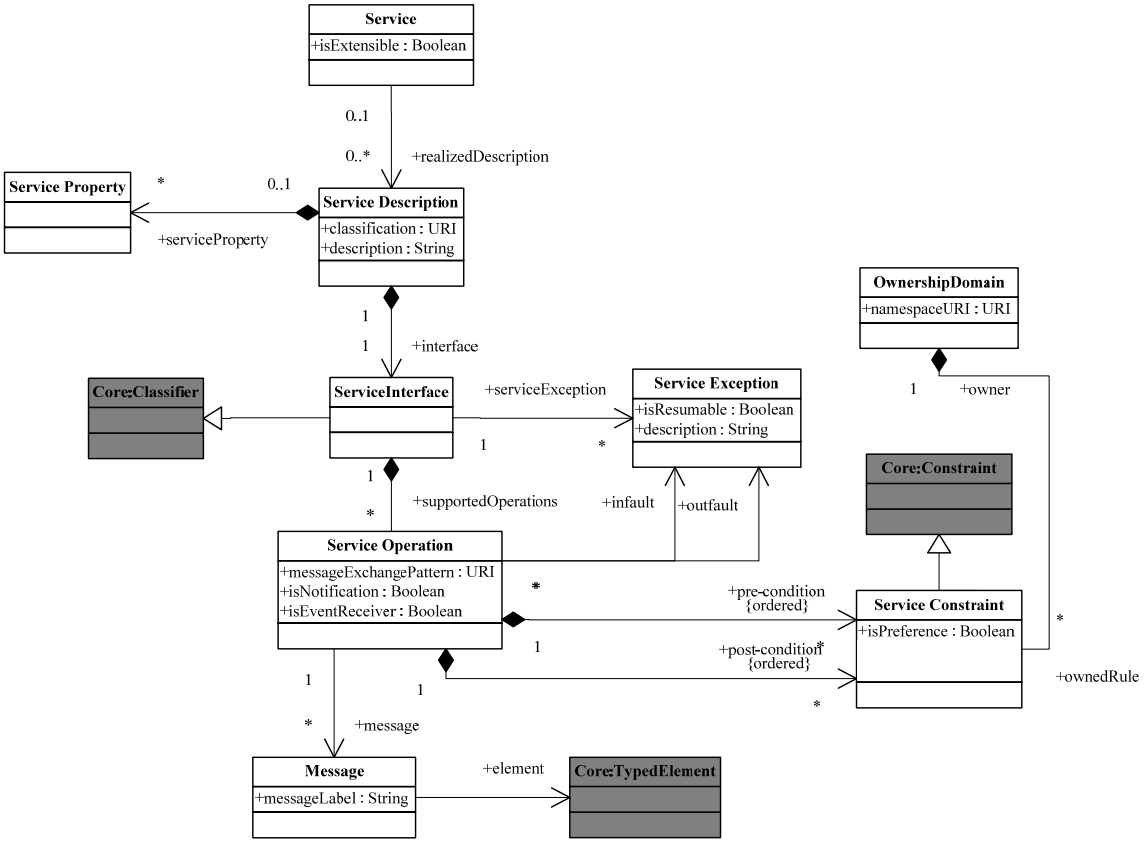


Fig 3. Service Capability View

Service Policy View

The service policy view (fig. 4) supports the specification of policies of the service participants. The policies complement the service capability by describing the non-functional enforceable constraints on a service. The constraints could either be technical constraints or business constraints. An example for technical policies would be security policy. The business policies could be pricing policy, loyality program policy etc. These are specified by concerned domain experts. The semantics for the service policy view is derived from WS-Policy [19] specification which is the W3C recommendation for expressing policy constraints on services. By following standards-based semantics, we hope to reduce the representation gap.

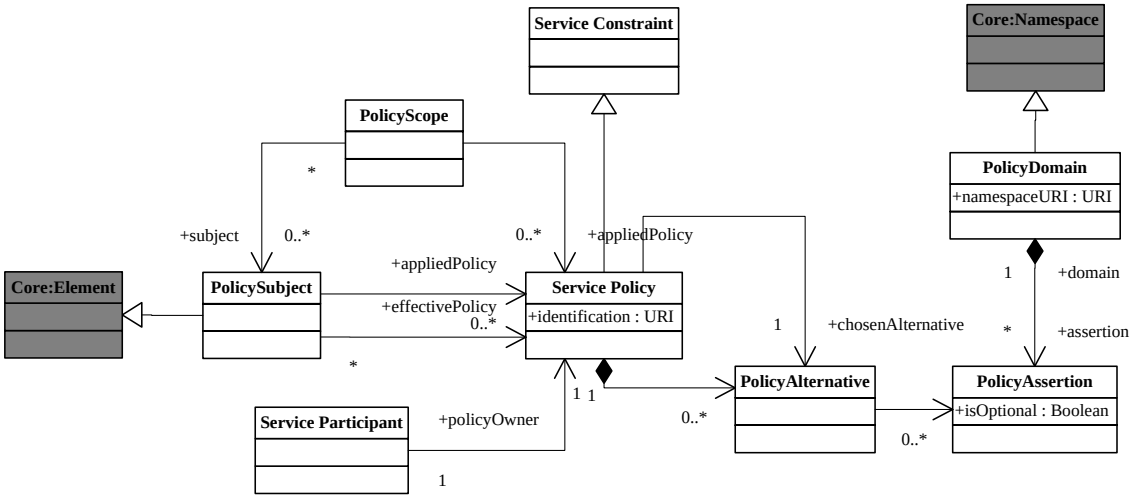


Fig 4. Service Policy View

Key Classes and Associations

Service Policy: A service policy defines a set of enforceable constraints which would be applied on a policy subject. It reflects the point of view of a service participant. It has a namespace URI attribute and extends the service constraint (from service capability view). A service participant is a policy owner of a service polocy.

Policy Subject: A policy subject represents the entity on which a policy is applied. A policy subject extends the *Core: Element.* Ownership Domain, Service, Service Interface, Service Operation, Message and Interaction Point (explained later) could be policy subjects on which a policy is applied. If a set of policies are applicable on a single policy subject, these are reconciled and represented as an ‘effective policy’.

Policy Scope: A policy scope represents a set of policy subjects on which a policy could be applied. It is a mechanism to group policy subjects together in order to apply the same policy on them. More than one policy could also be applied on the policy scope.

Policy Alternative: Each policy has a set of policy alternatives out of which at least one has to be honored. The policy alternative which is honored is called the ‘chosen alternative’.

Policy Assertion: A policy alternative has a set of policy assertions. Each of these assertions is typed by an assertion type which belong to domain policies of various domains – technical (security, reliability etc.) as well business (pricing, availability etc.)

Policy Domain: A policy domain represents a grouping of assertions belonging to a particular business domain such as pricing or availability or technical domains such as security, trust etc. A policy domain is identified by a name and a namespace URI.

Constraints

None

Service Realization View

The service realization view (fig. 5) helps to describe how services specified using the service definition, capability and policy views are realized. Service realization is done by IT Experts. Realization of a service could be either through implementation or through composition. Atomic services are realized through service provider implementations whereas composite services are realized through composition of existing services. These service providers could be existing IT assets or new implementations. Composition is achieved by service aggregators using known composition patterns and composition directives. Design-time composition directives enable dynamic composition decisions at execution-time.

Key Classes and Associations

Service: Service has an attribute 'isComposable' which determines if the service is composable. Also if a service is a composite service, then it is composed of many constituent services. Each composite service has an associated composition pattern.

Service Participant: A service participant represents a role played by a stakeholder in a services marketplace. Service Provider, Service Aggregator, Service Consumer and Service Mediator are different roles representing service participants. It extends the *Core: Classifier.*

Service Provider: A service provider provides the implementation for realization of an atomic service. A service provider could be an off-the-shelf component, a function module in a packaged application or even a stored procedure in a database. It could also be a new implementation. A service provider is a service participant.

Service Aggregator: A service aggregator aggregates different (constituent) services to provide a value-added composite service through service composition. A service aggregator is a service participant.

Composition Pattern: A composition pattern is a pattern which describes a structured assembly of constituent services to create a composite service. There are many patterns available in the literature [20] [21]. The composition pattern extends the *Core: NamedElement.*

Composition Directive: The composition directive represents a directive used for composing consituent services to create a composite service using the composition pattern. A composition directive is associated with a composition pattern. It extends the *Core: OpaqueExpression.*

Constraints

1. Only Composite services have a composition pattern associated with them.
2. Only Composite services have an aggregator associated with it.
3. The supported service of a service provider is always an atomic service

4. The supported service of a service aggregator is always a composite service
5. If one or more constituent services of a composite service is abstract, then the constituent service is also abstract.

Service Mediation View

In a loosely-coupled environment, mediators are needed to cope with inherent heterogenities. Service Mediators are used to re-purpose services to cater to a wider variety of user goals. Such a mediation is called process mediation. Service Mediation is also needed during service composition to support differences in service data and message schemas. Such a mediation is called data mediation. The Service Mediation View (fig. 6) helps in defining data or process mediation scenarios.

Key Classes and Associations

Service Mediator: A service mediator facilitates mediation for a service. Mediation could either be data or process mediation . The type of mediator is specified by MediatorType (data or process). The mediatorType attribute denotes the actual mediation. The service mediation provides a mediation service which supports the actual mediation.

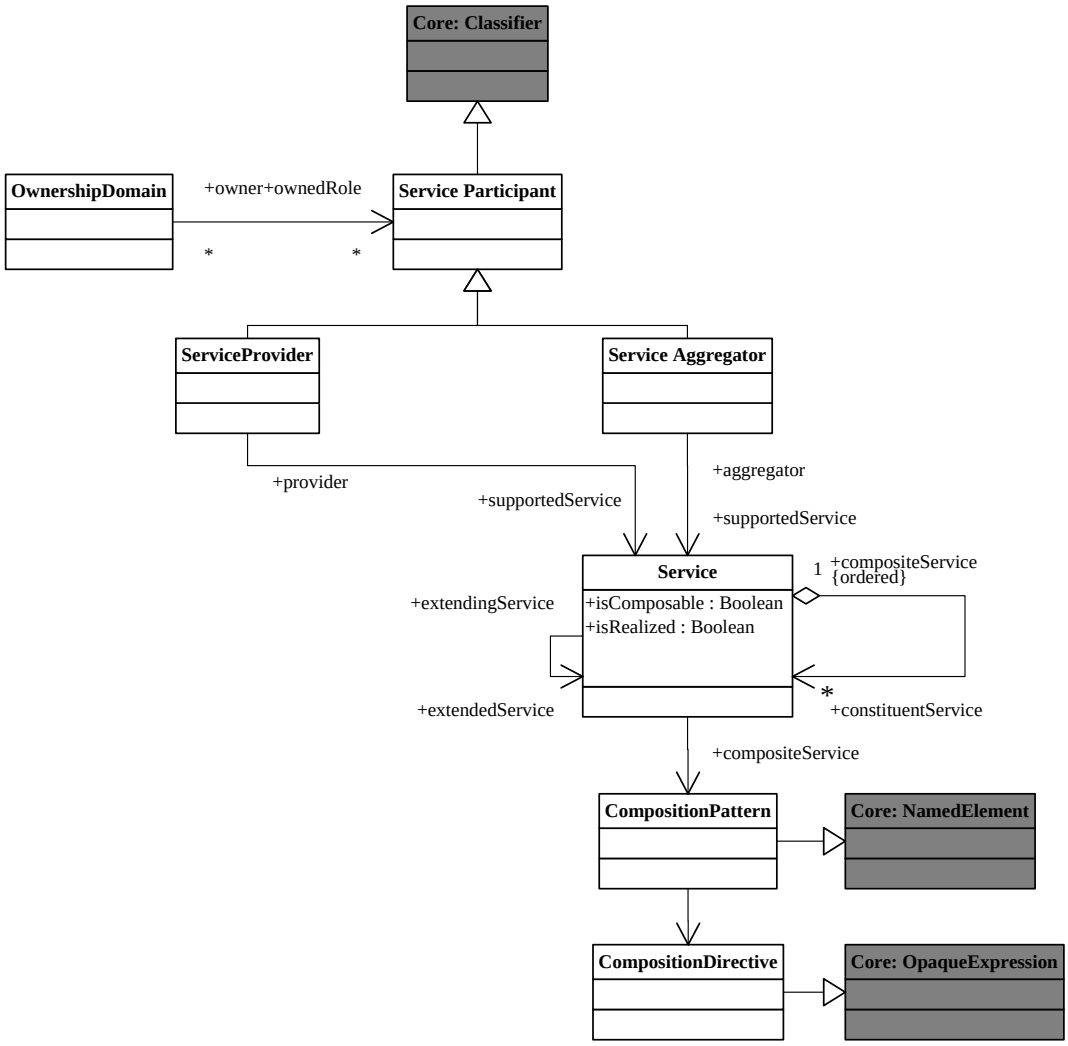


Fig 5. Service Realization View

Service: A service has an attribute 'isMediator' which signifies whether a service is a mediator or not. The mediator service can either mediate between a consumer and a service or between a service and another service. It also has another attribute 'isRealized' which determines whether the service has a realization.

Goal: The goal represents the goal (or need) of a service consumer. Each Service Consumer has associated goals (consumer goals). Each of these goals is satisfied by one or more services (satisfying services) and each service supports one or more goals. A goal has both pre- and post-conditions which have to be met if the goal has to be satisfied. Goal extends *Core: NamedElement.*

Service Mediation: Service Mediation represents the mediation between either two services (in a composition scenario) or between a service and an external service

consumer. A service mediator is associated with a mediator service which does the actual mediation. It extends the *Core: DirectedRelationship.*

Constraints

1. Abstract services are not realized (isRealized = false).

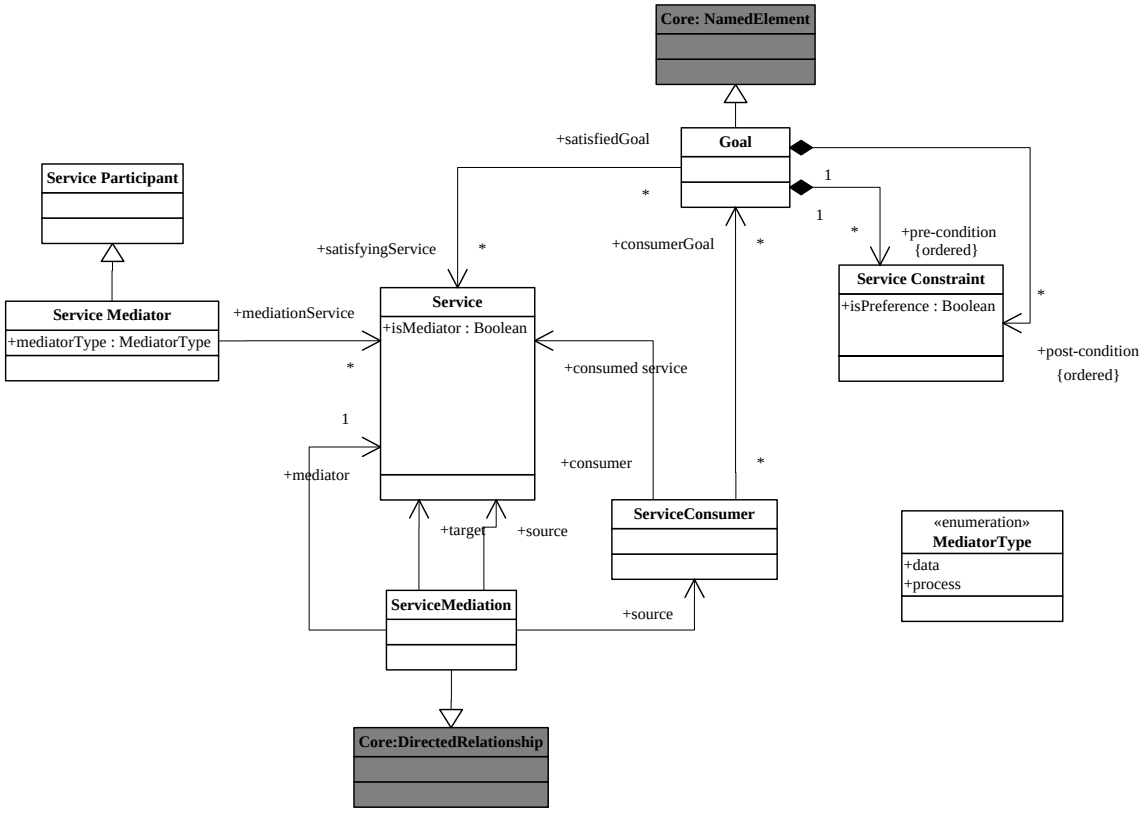


Fig 6. Service Mediation View

Service Deployment View

The service deployment view (fig. 7) helps to describe how realized and concrete services are deployed and how they could be invoked by external stakeholders.

Key Classes and Associations

Interaction Point: The interaction point defines an endpoint at which a service could be accessed by service consumers. It is uniquely determined by a location URI. Interaction point encapsulates the semantics of a channel through which a service is exposed (exposed service). The choice of a channel is represented by a BindingType i.e. logical channel type such as SOAP, HTTP etc. An ownership domain may have one or more

interaction points. Also a service could be exposed through different interaction points (end-points).

Service Invocation: Service invocation defines an invocation of a service through the interaction point by an external service consumer or another service (in a service composition scenario). It extends the *Core: Directed Relationship*. It also defines the mode of interaction i.e. Invocation Mode – whether the service invocation is synchronous or asynchronous. A service invocation could be either stateful (isStateful = true) or stateless.

Service: Service has an attribute 'isDeployed' which signifies if a service is deployed and has at least one interaction point.

Constraints

1. Abstract services will not have interaction points since they can not be deployed (isDeployed = false).

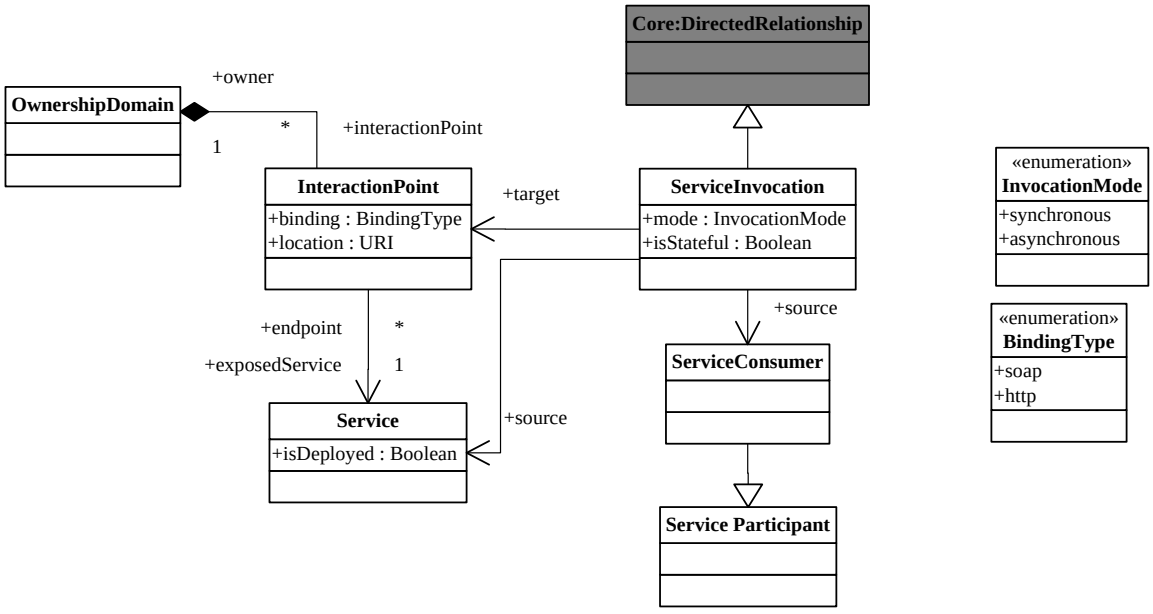


Fig 7. Service Deployment View

5 MODELING THE INTERNET AUCTIONS SCENARIO

In this section we would use the model views described in Section 4 to model the eBay Auctions Scenario. In fig 7, we use the service definition view to define the services in eBay® auctions. The eBay® auctions ownership domain owns atomic services like 'AuctionItem', 'BidForItem', 'RateBuyer', 'RateSeller' and composite services like 'BuyItNow' service. The ownership domains like 'Feedback' and 'Auction' are present

under the eBay® Auctions ownership domain. The Paypal® ownership domain owns the 'ProcessPayment' atomic service. The shipping service is provided by UPS® ownership domain through 'ShipItem'.

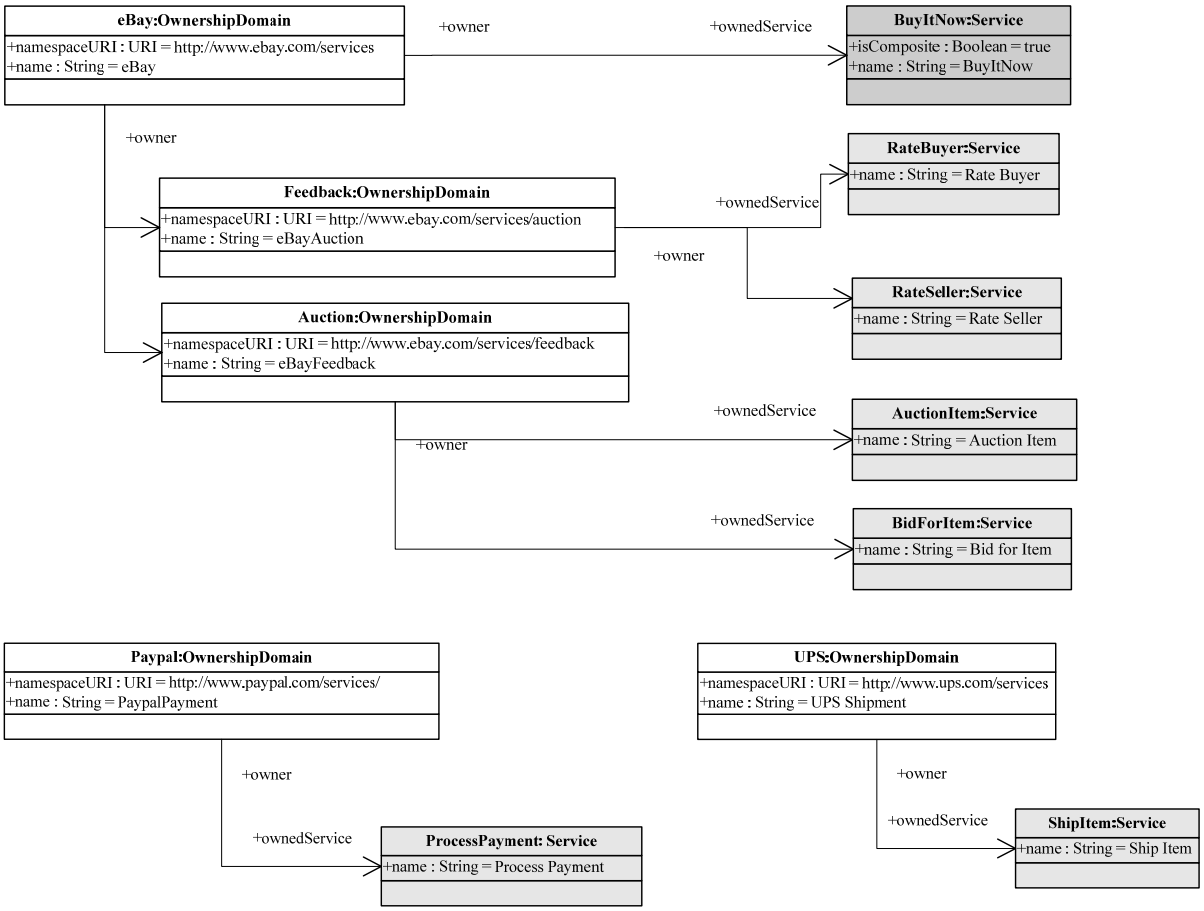


Fig 7. eBay Auctions Scenario – Service Definition

In fig 8, the 'AuctionItem' service capability is modeled using the service capability view. The realized service description denotes the classification. The scheme used in this case is the NACIS. The service is classified as a business-to-consumer (B2C) auction service. The service description has a service interface with an associated service exception. The 'AuctionNotPermitted' exception denotes the business contract violation of trying to auction an item which is prevented from being auctioned. The service interface has a service operation 'AuctionItem' which follows the 'request-response' message exchange pattern. Since the pattern supports an 'in' as well as an 'out' message, we have defined both the messages. The 'in' message encapsulates item name, item description, minimum bid and bid closing date.

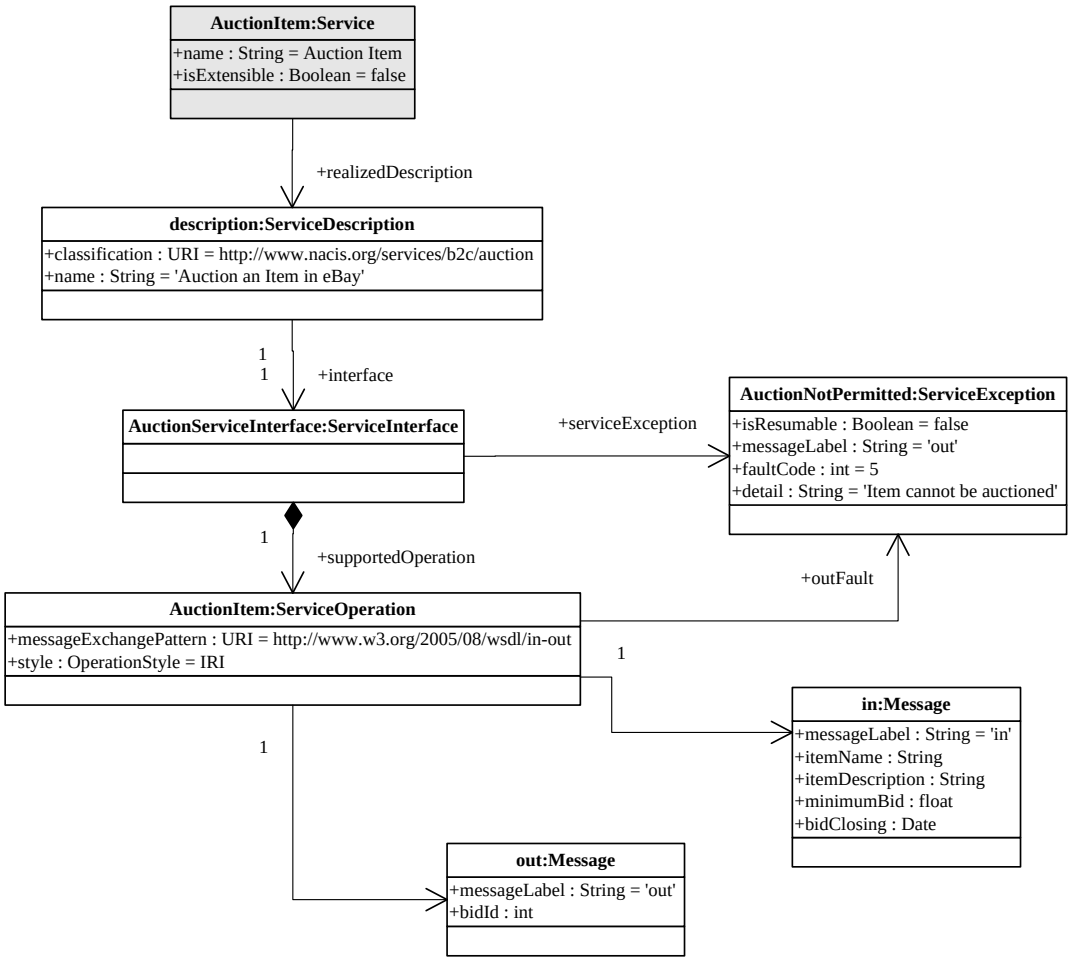


Fig 8. Auction Item – Service Capability

In fig 9, the realization of the ‘AuctionItem’ service is modeled. The auction manager is a service provider that realises the atomic services ‘Auction Item’ and ‘BidForItem’ service. The ‘BuyItNow’ service is a composite service whose constituent services are ‘ProcessPayment’ and the ‘ShipItem’. The ‘BuyItNow’ service composes these services using the sequential composition pattern.

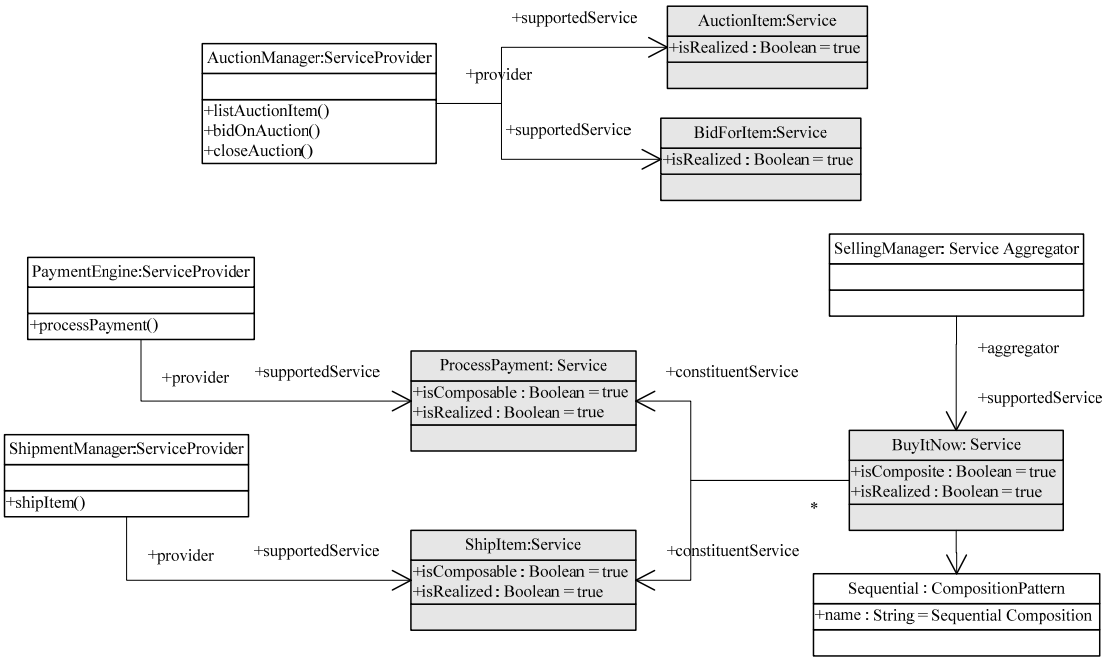


Fig 9. eBay Services – Service Realization Model

In fig 10, the deployment scenario of the 'Auction Item' service is modeled. The 'AuctionItem' service is exposed through an interaction point 'Auction'. The transport binding for this endpoint is SOAP. The location URI is also mentioned through which the service can be invoked 'synchronously'. The service consumer is a 'Seller' who belongs to the public domain.

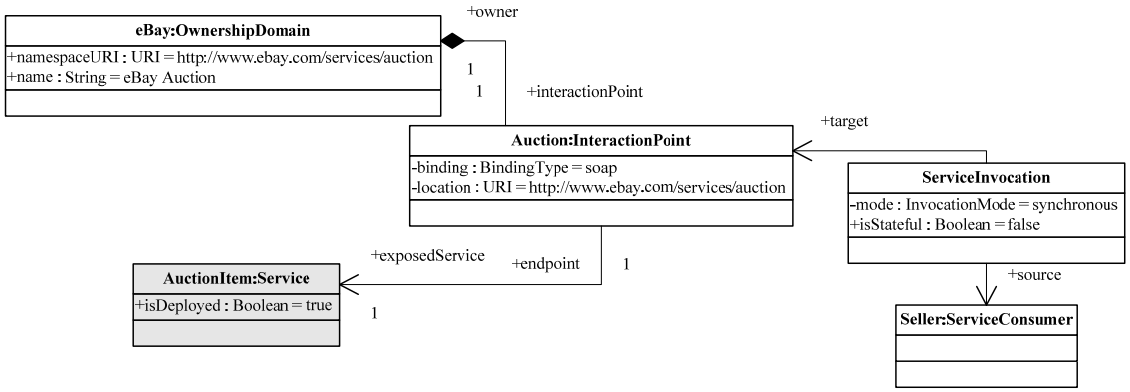


Fig 10. Auction Item – Service Deployment Model

In fig 11, the proxy bidding scenario is modeled. In the proxy bidding scenario, the bids on a particular item are revised automatically based on user-specified criteria. The goal of the service consumer – the bidder is to place proxy bids. The proxy bidding service is a process mediation service that uses the existing service 'BidForItem'.

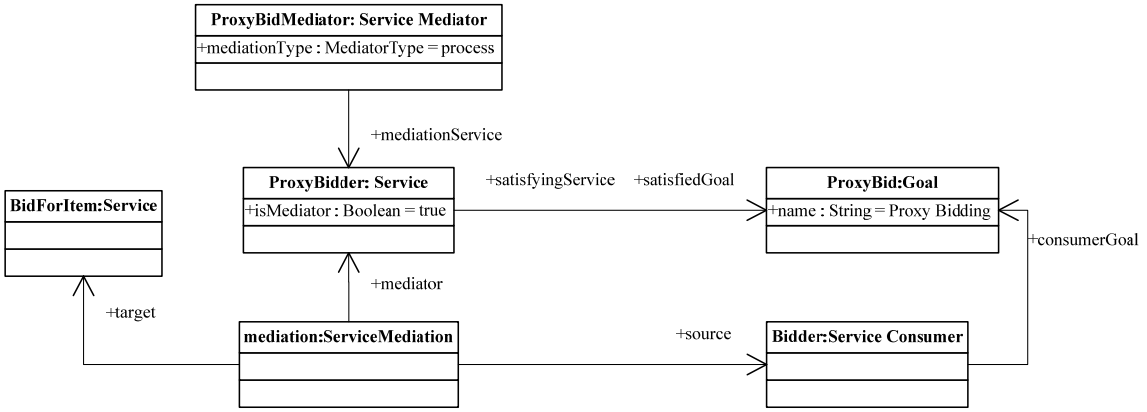


Fig 11. Proxy Bidding – Service Mediation Model

In fig 12, the policy for the auction service is modeled. In the Service Policy View (fig. 8) the security policy is defined for the 'Auction Item' service. The IT expert refines the mechanisms for the security policy. There are 2 alternatives modeled, Alternative 1 supports a X.509 digital certificate and a HTTPS transport. Alternative 2 supports a Kerberos certificate and HTTPS transport. All these assertions belong to the Security policy domain.

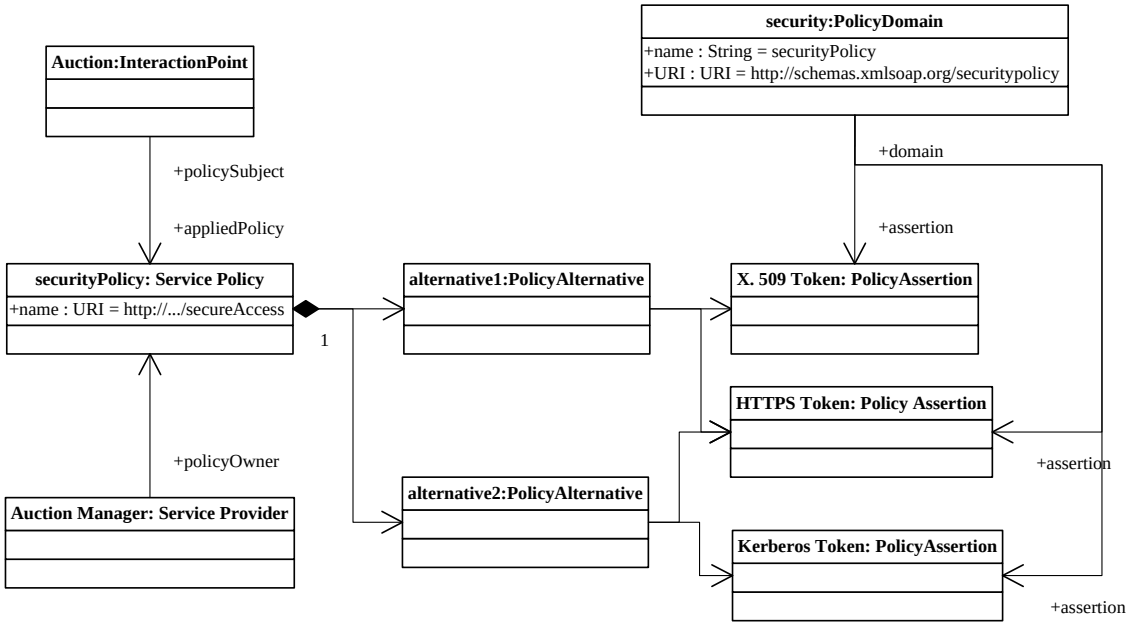


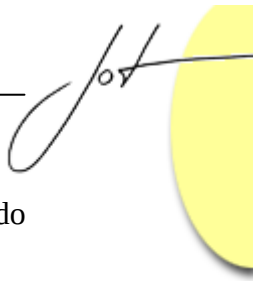
Fig 14. Auction Service – Service Policy Model

6 CONFORMANCE

We wish to ensure compliance with the OASIS Reference Model for Service-Oriented Architectures [22]. We present the state of our compliance to the reference model's conformance guidelines (Section 4 of [22]) below:

1. Have entities that can be identified as services as defined by this Reference Model
   We have a first-class model entity 'Service' in our metamodel. Service represents a set of capabilities provided by a service provider (or a service aggregator) which meets the goals (needs) of service consumers.
2. Be able to identify how visibility is established between service providers and consumers
   A service consumer could become aware of a service provider and its capabilities on offer through a service description (awareness). However we do not address disovery and advertising capabilities as yet. Service providers and service consumers interact through an interaction point (reachability).
3. Be able to identify how interaction is mediated
   The interaction is mediated through the understanding provided by the service description. The message exchange patterns of service operation dictate the sequence of communication between the provider and the consumer.
4. Be able to identify how the effect of using services is understood
   Given that the pre-conditions and policies are met, the post-conditions on a service speration specify the real-world effects of invoking the service operation.
5. Have descriptions associated with services
   Service description (containing the service interface, associated operations and properties) and the service policy provide description about choosing and using a particular service.
6. Be able to identify the execution context required to support interaction
   Though we have the infrastructural elements such as service description, service policy we do not completely address all requirements of execution context to support interaction between the providers and consumers.
7. It will be possible to identify how policies are handled and how contracts may be modeled and enforced
   Our service policy view completely addresses modeling of policy alternatives, assertions for a service. Enforcement of policy is outside the scope of this metamodel. Modeling support for service contracts is still missing.

We also compare our model to W3C's Web Services Architecture [23]. In Table 1. we present a comparison of the concepts present in the web services architecture with the concepts in our metamodel. The web services architecture represents the concepts and their relationships as concept maps. Whereas we have a formal model based on MOF2. Since our focus is on early-stage design, certain concepts (ex. message body and header)



are not present in our model. Also we do not view services are resources and hence we do not have a comparable model to the resource-oriented model.

Web Services Architecture	**Services Metamodel**
Service Oriented Model	Service Definition / Service Capability View
Service	Service
Service Description	Service Description
Service Interface	Service Interface
Person / Organization	Ownership Domain
Provider Agent	Service Provider
Requestor Agent	Service Requestor
Service Intermediary	Service Mediator
Policy Model	Service Policy View
Policy	Service Policy
Policy Description	Service Policy
Domain	Domain Assertion
Permission / Obligation Guard	Policy Assertion
Resource Oriented Model	None
Service (is a Resource)	(Service is not viewed as a resource in our model)
Message Oriented Model	Service Capability View
Message	Message
Message Exchange Pattern	Service Operation's Message Exchange Pattern

Table 1. Related Concepts in the Web Services Architecture

7 RELATED WORK

Model-driven development of services is still in the nascent stage. Model-driven development of web-services is addressed in [24][25][26]. The RFP-UMPS is an effort by OMG to consolidate existing approaches into a consistent metamodel and UML2 profile for modeling services. There are existing UML-based approaches to modeling services. [26] uses UML class diagrams to model services. In this approach Service is not viewed as a first-class modeling entity. UML Collaboration diagrams have been used extensively to model behavioral-aspects such as service collaboration and compositions [28] [29].

There are also other efforts to provide support for services modeling through light-weight extensions to UML through Profiles [30] [31] [32]. All these efforts provide a direct mapping between WSDL 1.1 elements and their model elements. Also they are based on the UML1.x standards. UML-profiles for services and SOA are proposed by [33] [34]. An UML 2.0 Profile for Software Services [35] is proposed by IBM. In this profile, a service is restrictively modeled as a Port of a UML Composite class. The service realization mechanisms are only through implementation by components.

Composition as a realization mechanism is not supported. The profile does not support modeling of policies and mediation. Although Service is a first-class modeling entity, it is tightly associated with a Component. In contrast, in our services metamodel services are truly first-class modeling entities. Modeling of realization mechanisms such as implementation and composition are supported. Our service metamodel also supports modeling non-functional aspects of services through service properties and policies. We also provide support for deployment and mediation of services.

UML-profile for distributed object computing (EDOC) [36] facilitates modeling of enterprise systems but does not provide means to model services. The UN/CEFACT's Unified Modeling Methodology (UMM) [37] provides a standard way for business processes and information modeling for e-Commerce. An UML-profile for B2B e-commerce is presented in [38]. [39] proposes a framework used to derive SOA models from already existing enterprise models. Our services metamodel could complement these approaches and act as the foundation for a model-based service repository.

Apart from UML-based modeling approaches, there are other approaches which aid modeling of services. [40] provides a formal-model of services with a theoritical foundation for specifying services and service composition. The Web Services Modeling Framework (WSMF) [41] defines conceptual entities for service modeling. Web-Service Modeling Ontology (WSMO) [42] has its foundations in WSMF but it defines a formal ontology to semantically describe web services. The Web Services Modeling Language (WSML) [43] provides a formal syntax for WSMO based on different logical formalisms.

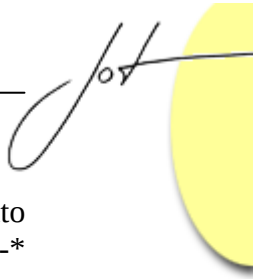
8 CONCLUSIONS AND FUTURE WORK

In this paper we presented a services metamodel with six model views to model different perspectives of services development. These model views have a formal foundation based on MOF2. They support different stakeholders such as business experts and the IT experts to model services during early-stage services design. The metamodel derives its foundations from technical specifications like WSDL 2.0, WS-Policy and WSMF since our focus is on web-based electronic services. We have used our services metamodel to model a fictitious eBay® auctions scenario. Through this modeling exercise, we have demonstrated how different facets of services such as service description, realization, mediation and deployment could be modeled using our services metamodel.

Our future research would be in the following directions:

Defining a UML2 profile for our services metamodel in order to leverage existing tool support.

Investigating the use of the the non-functional property modeling framework (NFP) from the UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) [44] for modeling service properties.



Using MOF Query, View and Transform (QVT) [45] to transform these models into executable specifications such as WSDL, WS-Policy and the associated WS-* specifications for web services.

REFERENCES

[1] Papazoglou, M.P. & Georgakopoulos, D.: Service-Oriented Computing. In Communications of the ACM, 46(10): 25-28, October 2003.

[2] Moore, G. Living on the Fault Line: Managing for Shareholder Value in the Age of the Internet, HarperBusiness, New York, 2000.

[3] OASIS. Web services security: SOAP message security. OASIS TC Working Draft 10, Feb. 2003.

[4] Web Services Transaction (WS-Transaction) 1.0. Available at: www-106.ibm.com/developerworks/library/ws-transpec.

[5] Web Services Coordination (WS-Coordination) 1.0. Available at: www-106.ibm.com/developerworks/library/ws-coor.

[6] eBay, eBay Developer Program, Retrieved April 26, 2007, from http://developer.ebay.com/

[7] E. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S.Thatte, D. Winer, Simple Object Access Protocol (SOAP) 1.1, May 2000. Available at http://www.w3.org/TR/SOAP

[8] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft 26-March-2004. Available at: http://www.w3.org/TR/2004/WD-wsdl20-20040326/.

[9] R. Fielding, Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

[10] Bray, J. Paoli, C. M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml.

[11] D.Frankel,J.Greenfield, Model-Driven Development: Automating Component Design, Implementation, and Assembly. John Wiley, 2002

[12] Adaptive Ltd, Ceira Technologies Inc., Compuware Corporation, Data Access Technologies Inc., DSTC, Gentleware, Hewlett-Packard, International Business Machines,IONA Technologies, MetaMatrix, Rational Software, Softeam, Sun Microssystems, Telelogic AB, Unisys, und WebGain: Meta Object Facility (MOF) 2.0 Core Proposal. April 2003. ad/2003-04-07.

[13] OMG: UML 2.0 Infrastructure Final Adopted Specification, OMG document ad/03-09-15, http://www.omg.org/docs/ad/03-09-15.pdf (2003)

[14] OMG, UML Profile and Metamodel for Services (UPMS), Request For Proposal. OMG Document, reference 'soa/2006-09-09', September 2006

[15] Plessis, Deon. "An Ebay Auction Businesses With A Twist." EzineArticles 20 April 2007. 27 April 2007. Available at: http://ezinearticles.com/?An-Ebay-Auction-Businesses-With-A-Twist&id=534557.

[16] SAP, mySAP Supplier Relationship Management, Retrieved April 26, 2007, from http://www.sap.com/solutions/business-suite/srm/index.epx

[17] North American Industrial Classification System (NAICS)-United States, 1997. http://www.census.gov/epcd/www/naics.html

[18] W3C Recommendation: Web Services Description Language (WSDL) Version 2.0 Primer.

[19] S. Bajaj, et al. Web Services Policy Framework 1.2 Spec. BEA, IBM, Microsoft, SAP, Sonic, VeriSign, Mar. 2006. download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf

[20] B. Benatallah, M. Dumas, M.-C. Fauvet, and F. Rabhi. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi, editors, Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, UK, Nov 2002

[21] J.Yang, M.Papazoglou, Web Component: A Substrate for Web Service Reuse and Composition, Proceedings of the 14th International Conference on Advanced Information Systems Engineering, p.21-36, May 27-31, 2002

[22] OASIS Reference Model for Service Oriented Architecture 1.0, Committee Specification 1, 2 August 2006.

[23] The W3C Web Services Architecture working group, public draft, August 2003. Available at: http://www.w3.org/TR/2003/WD-ws-arch-20030808/

[24] Gronmo, R., Skogan, D., Solheim, I., Oldevik, J.: Model-driven Web services development. In: EEE'04, pp. 42-45. IEEE, Los Alamitos (2004)

[25] Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F.: Applying MDA Approach for Web Service Platform. In: Proceedings 8th International Enterprise Distributed Object Computing, pp. 58-70 (2004)

[26] A Platform Independent Model for Service Oriented Architectures. Gorka Benguria, Xabier Larrucea, Brian Elvesæter, Tor Neple, Anthony Beardsmore, Michael Friess. I ESA Conference 2006. Bordeaux. Enterprise Interoperability New Challenges and Approaches, 2007.

[27] Baresi, L., Heckel, R., Thöne, S., Varró, D.: Modeling and validation of service-oriented architectures: application vs. style. In: Proceedings of the 11th ACM

SIGSOFT Symposium on Foundations of Software Engineering 2003, ESEC/FSE, pp. 68-77 (2003)

[28] Skogan, D., Gronmo, R., Solheim, I.: Web Service Composition in UML. In: Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference (EDOC), IEEE Computer Society Press, Los Alamitos (2004)

[29] Sanders, R., Castejón, H., Kraemer, F., Bræk, R.: Using UML 2.0 Collaborations for Compositional Service Specification. In: Proceedings of the 8th International Conference of Model Driven Engineering Languages and Systems, pp. 460-475 (2005)

[30] D.Frankel, J.Parodi. Using Model-Driven Architecture to Develop Web Services, IONA Technologies , 2ed. April 2002.

[31] J. Bezivin, S. Hammoudi, D. Lopes, F. Jouault, An Experiment in Mapping Web Services to Implementation Platforms, Atlas Group, INRIA and LINA University of Nantes, Research Report, March 2004

[32] Bordbar, B., and Staikopoulos, A. Automated Generation of Metamodels for Web Service Languages. In Proc. of Second European Workshop on Model Driven Architecture (MDA) (2004)

[33] A UML profile for service oriented architectures. Rafik Amir, Amir Zeid. OOPSLA 2004. ISBN:1-58113-833-4

[34] Towards a UML Profile for Service-Oriented Architectures. Reiko Heckel, Marc Lohmann, and Sebastian Thöne. MDAFA-2003

[35] S. Johnston, UML 2.0 Profile for Software Services, IBM Developer Works.

[36] Enterprise Collaboration Architecture: (ECA) Specification. Version 1.0. formal/04-02-01 (February 2004), http://www.omg.org/docs/formal/04-02-01.pdf

[37] Hofreiter, B., Huemer, C., Naujok, D.: UN/CEFACT's Buisness Collaboration Framework- Motivation and Basic Concepts. In: Proceedings of the MKWI (2004)

[38] UN/CEFACT's Modeling Methodology (UMM): A UML Profile for B2B e-Commerce B. Hofreiter, C. Huemer, P. Liegl, R. Schuster, M. Zapletal 2nd International Workshop on Best Practices of UML (BP-UML'06) @ Int'l Conf. on Conceptual Modeling (ER 2006), Springer LNCS, Tucson (Arizona, USA), November 2006

[39] Architecting SOA Solutions from Enterprise Models: A model driven framework to architect SOA solutions from enterprise models Xabier Larrucea, Gorka Benguria. ICEIS 2006. "Enterprise Information Systems VI", 2006, XIV, 326 p., Hardcover, ISBN: 1-4020-3674-4, 2006

[40] Broy, M., Krüger, I.H., Meisinger, M.: A formal model of services. ACM Trans. Software Engineering Methodology 16 (February 2007)

[41] Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. Electronic Commerce:Research and Applications 1 (2002) 113-137

[42] H. Lausen, A. Polleres, and D. Roman (eds.). Web Service Modeling Ontology (WSMO). W3C Member Submission 3 June 2005, 2005. Available at: http://www.w3.org/Submission/WSMO/.

[43] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Technical report, WSML Working Draft, http://www.wsmo.org/TR/d16/d16.1/v0.2/, March 2005

[44] Object Management Group, UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE), RFP. 2005. OMG document: realtime/05-02-06.

[45] MOF QVT - QueryViews/Transformations. OMG doc. ptc/2005-11-01, Nov. 2005.

About the authors

Harshavardhan Jegadeesan works in the Research & Breakthrough Innovation group of SAP Labs, India. His areas of interest include enterprise service-oriented architectures, enterprise systems and business process platforms. He can be reached at harshavardhan.jegadeesan@sap.com

Sundar Balasubramaniam is an Associate Professor of Computer Science at BITS, Pilani. He is also the Group Leader (a.k.a. Head of Department) of the Computer Science & Information Systems department. He can be reached at sundarb@bits-pilani.ac.in